I MICROPROCESSORI E LE LORO APPLICAZIONI:

SC/MP

ALDO CAVALCOLI
VALERIO SCIBILIA

JACKSON
ITALIANA
EDITRICE

I MICROPROCESSORI E LE LORO APPLICAZIONI: SC/MP

Autori

ALDO CAVALCOLI

VALERIO SCIBILIA

Mipro s.r.l.

JACKSON
ITALIANA
EDITRICE

Piazzale Massari, 22
20125 Milano

PREFAZIONE

Un volume della serie "I Microprocessori e le loro applicazioni" dedicato al microprocessore SC/MP della National Semiconductor e, come seconda sorgente, della Signetics.
Si tratta di un microprocessore a caratteristiche semplici e di facile applicazione, che fornisce materia di studio e sperimentazione al maggior numero possibile di potenziali lettori.
Le applicazioni presentate, infatti, possono essere, per la maggior parte, realizzate senza l'ausilio di una particolare e costosa attrezzatura e sono comunque indirizzate alla soluzione di classici problemi che si presentano normalmente nella progettazione con sistemi a microprocessore.

SOMMARIO

INTRODUZIONE

Questa pubblicazione si inserisce in una serie di monografie aventi come argomento i microprocessori.
Con questa iniziativa la Jackson Italiana Editrice, in collaborazione con la MIPRO, vuole colmare una lacuna nell'editoria tecnica in un momento in cui inizia a farsi sentire, da parte sia di progettisti che di responsabili di scelte aziendali, l'esigenza di avere a disposizione degli strumenti di lavoro e di consultazione riferiti alla tecnologia dei microprocessori.
Questo volume, contiene sia parti tecniche che pratiche, in modo da permettere alla più vasta gamma di lettori di trarre la materia di studio di cui necessitano. Del resto è finito il periodo in cui parlare di microprocessori poteva essere considerata una moda, ora è una reale esigenza tecnica; non ci si chiede più cosa sono questi nuovi componenti, ma come possono essere utilizzati.
Rispondere a questa domanda significa essere in grado di inquadrare la tecnologia dei microprocessori in realtà aziendali diverse, specializzando le problematiche generali di utilizzo alle singole aree in cui si opera.
L'efficienza di questa operazione è chiaramente funzione dell'esperienza che si possiede; mai come nel caso dei microprocessori si è dimostrata fondamentale la capacità di individuare e gestire le proprie reali necessità di progetto e di produzione in modo autonomo.
A nostro parere è importante abbandonare l'usuale metodologia di acquisizione di know-how: far riferimento ad applicazioni già realizzate, preferibilmente all'estero, trarre da queste delle linee generali e modellare le proprie esigenze su quanto altri hanno creato, condizionando di fatto la propria autonomia progettuale, con conseguente annullamento di quelle potenzialità che sono proprio alla base della estrema flessibilità dei microprocessori.
Quanto detto è confortato dalla situazione italiana nel campo della informatica, situazione di totale dipendenza dalle grandi case costruttrici, che indica non solo i mezzi per risolvere i problemi, ma anche i problemi da risolvere con mezzi già ideati e collaudati.
Potrebbe succedere la stessa cosa con i microprocessori: non per nulla una delle caratteristiche fondamentali dei microprocessori, causa anche di una certa diffidenza da parte dei progettisti, è la presenza del software, la necessità di indicare al microprocessore, con programma, le operazioni da eseguire per realizzare le funzioni volute.
Attualmente le case produttrici di microprocessori, oltre a fornire il componente, offrono tutta una serie di strumenti di supporto che vanno dal sistema di sviluppo completo microcomputer a librerie più o meno sofisticate di software: è inevitabile che in questa definizione di supporto si segua la strada già tracciata dai costruttori di grossi e medi sistemi di elaborazione dati.
Fino a che punto l'utente può condizionare il costruttore di microprocessori ed ottenere quello di cui ha realmente bisogno? Chiaramente nella misura in cui ha le idee chiare e la conseguente capacità di effettuare scelte autonome.
Nella realtà tecnologica attuale non sussistono le condizioni per cui questo si realizzi, quindi tutto va visto in prospettiva, ma è importante vederlo e soprattutto acquisire il concetto per cui ogni azienda ha i suoi problemi e, se sa risolverli con le collaudate tecnologie presenti, deve saperlo fare anche con i microprocessori,il cui utilizzo appare sempre più inevitabile.
Tutto il discorso si chiude sulla necessità di esperienza o quanto meno di conoscenza di base, non passivamente acquisite, che permettano un approccio critico alla tecnologia dei microprocessori, dove per approccio critico intendiamo la capacità di sapere filtrare il know-how altrui con le proprie esigenze attuali e future, autonomamente definite.
Gli autori ringraziano l'organizzazione MIPRO per l'aiuto fornito. Ringraziano altresì l'organizzazione NATIONAL SEMICONDUCTOR in Italia per il materiale messo a disposizione e per la collaborazione prestata. In particolare ringraziano: l'Ing. Caccia, l'Ing. Pinto, il Sig. Altieri.

PARTE I

Descrizione tecnica del microprocessore SC/MP

Capitolo 1°

1.1 INTRODUZIONE ALLO SC/MP

Il microprocessore SC/MP è una CPU (Central Processing Unit) a singolo chip economica, ma con ottime prestazioni.
Lo SC/MP è posto in un contenitore dual in-line a 40 pin; il layout del chip è mostrato in figura 1-1.
La figura 1-2 mostra invece l'architettura della CPU ed i segnali presenti ai vari pin del chip.
Saranno qui sotto elencate alcune caratteristiche dello SC/MP ed i conseguenti vantaggi per l'utilizzatore.

Vantaggi

* I programmi applicativi sono facili da scrivere e sono efficienti per quanto riguarda l'utilizzo della memoria.
* Supportato da un software di sviluppo completo.
* Alta affidabilità del sistema completo, legata all'affidabilità dei componenti LSI.
* Basso numero di componenti e quindi basso costo del sistema finito.
* Design del sistema finale molto semplice.

Caratteristiche tecniche dello SC/MP

* Cinque modi di indirizzamento della memoria e delle periferiche.
— Relativo al Program Counter.
— Immediato.
— Indicizzato.
— Auto-Indicizzato.
— Implicito.
* Quattro registri puntatori per l'indirizzamento.
— Costo ridotto per la formazione degli indirizzi.
— Possibile realizzare subroutines interne ad altre subroutines.
— Quarantasei tipi di istruzioni, a singolo byte ed a doppio byte.
— Struttura interrupt controllata via software.
— Input/output seriali controllati via software.
* Editor.
* Assembler.
* Caricatori.
* Debug.
* Basso numero di componenti presenti nel sistema.
* Basso consumo.
* Microprocessore a singolo chip (DIP a 40 pin).
* Generatore di clock ed oscillatore interni.
* Usa memorie standard.
* Utilizza componenti standard per l'interfacciamento con le periferiche.
* Operazioni statiche.
* Bus di indirizzamento di 12 bit statico tri-state.
* Interfacciamento diretto con memorie standard sino a 65k bytes.
* Bus bidirezionale di 8 bit tri-state per i dati e le informazioni di controllo.
* Linee seriali di input/output.
* Input/output compatibile TTL e CMOS.
* Tre flag in output controllati da software
* Due linee di test.
* Interfacciamento semplice e mediante componenti standard.
* I segnali di controllo per l'Accesso Diretto in Memoria (DMA) sono già disponibili.
* SC/MP è supportato da:
— Sistemi di sviluppo.
— Schede applicative premontate.
— Documentazione completa con molti esempi pratici.

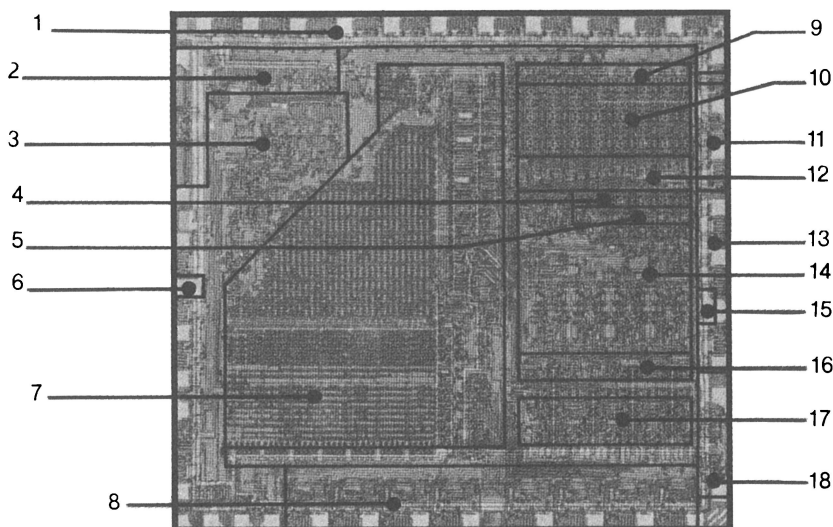



Figura 1-1. Layout del chip SC/MP

1 - Buffers degli indirizzi.
2 - Oscillatore e generatore di Clock.
3 - Logica di controllo di input/output.
4 - Accumulatore.
5 - Registro Extension.
6 - VGG.
7 - PLA (Programmable Logic Array)
8 - Buffer dei dati di input/output.
9 - Registro di indirizzamento della memoria.
10 - Program Counter e registri puntatori 1,2 e 3.
11 - Input seriale/Output seriale.
12 - Logica di scambio dati tra i bus interni.
13 - Output dei flags.
14 - ALU (Arithmetic Logic Unit).
15 - VSS.
16 - Registro degli stati.
17 - Registro delle istruzioni e registro dei dati di input/output.
18 - Ingressi di sense.

1.2 APPLICAZIONI DI SC/MP

SC/MP può essere utilizzato praticamente in tutte le applicazioni. Il chip può essere usato in una configurazione minima che comprenda alcuni switches per il controllo, una memoria read-only contenente il programma ed alcuni indicatori luminosi per la visualizzazione. D'altra parte, un sistema massimo può includere delle periferiche di ingresso e di uscita, memoria sia di tipo read/write che read-only ed un pannello di controllo completo.
Qualunque sia il vostro sistema, voi potete scoprire come SC/MP ed i prodotti che gli sono di supporto, possiedono tutto ciò che potete desiderare per sviluppare, testare e realizzare il vostro sistema.
Sono qui elencate alcune possibili applicazioni di SC/MP. Esse però non comprendono tutti i vantaggi, le configurazioni e gli usi possibili; mostrano solamente come delle applicazioni di SC/MP siano realmente a basso costo e possano risolvere i vostri problemi di controllo.

— Alcune possibilità d'uso

Sistemi di test e controllo di strumentazione.
Controllo di macchine utensili.
Piccole macchine contabili.
Sistemi di trattamento dati.
Sistemi didattici.
Sistemi a più processori.
Controllori di processo.
Controllo di terminali.
Controllo del traffico.
Strumentazione di laboratorio.
Giochi elettronici sofisticati.
Controllo di automobili.

— Alcune caratteristiche

Interfacciamento molto semplice con le periferiche.
Molti modi di indirizzamento.
Interfacciamento diretto con bus tri-state di indirizzamento e dei dati.
Accesso a tutta la memoria disponibile mediante quattro registri puntatori.
Input/Output sia seriale che parallelo.
Possibilità di indirizzare sino a 65.536 bytes di memoria.
Tre flags e due sense.
Interrupt controllato via software.
Aritmetica ad 8 bits sia binaria che BCD.
Oscillatore e generatore di clock interno.
Set di istruzioni di tipo general-purpose.
Documentazione, assistenza, software, informazioni applicative ed altro.
Una famiglia completa di prodotti di supporto.
Compatibilità TTL e CMOS.

— Alcuni vantaggi

Nessuno spreco di software o di hardware, è necessario acquistare solamente ciò che è indispensabile.
Si ha un sistema a building-block; è quindi possibile espanderlo, quando necessario, ad un costo minimo.
Il sistema finale non nasce obsoleto.

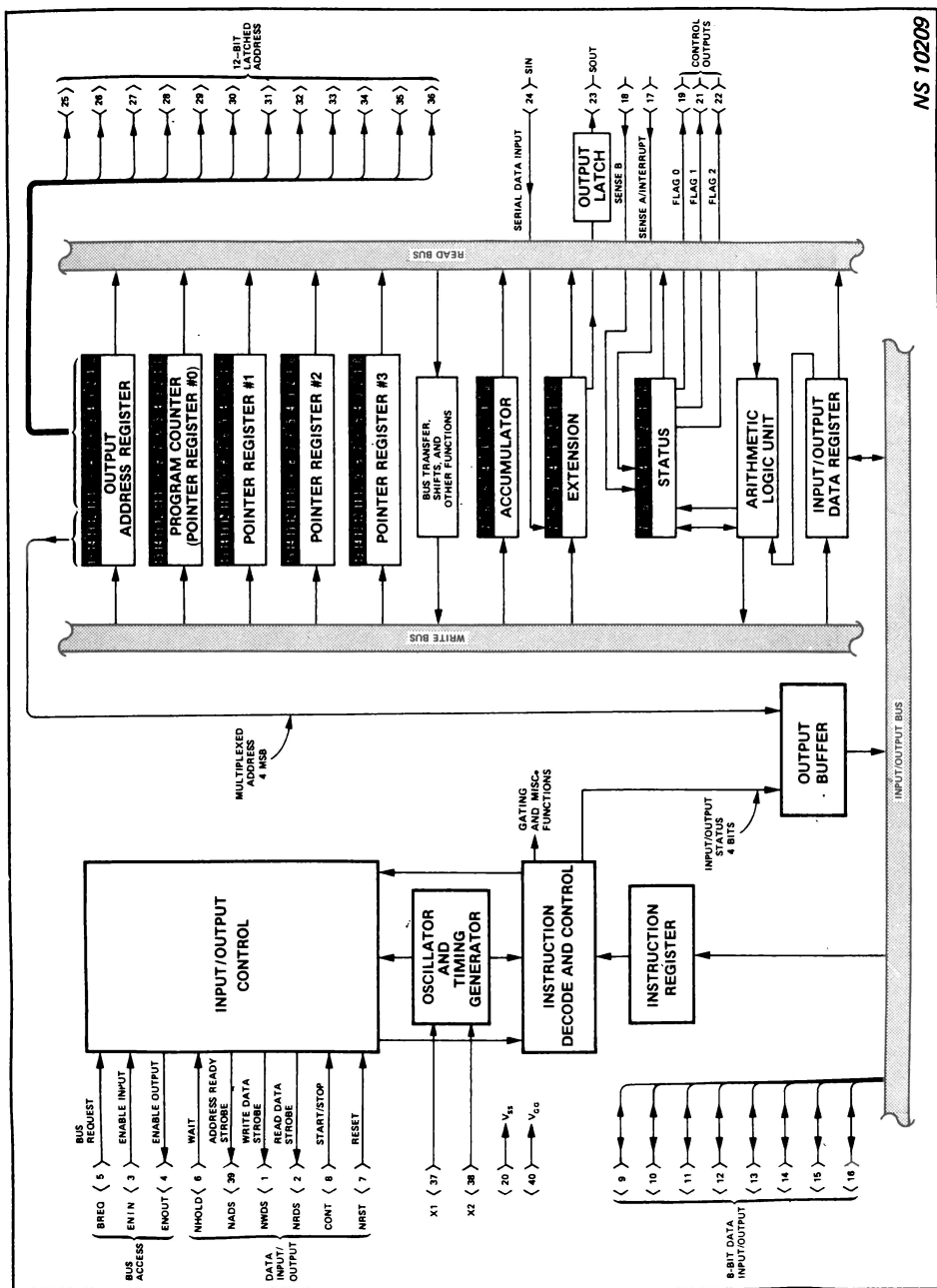



Figura 1-2. Architettura di SC/MP.

1.3 ARCHITETTURA DI SC/MP.

1.3.1 Considerazioni hardware

SC/MP possiede tutte le caratteristiche di un microprocessore di tipo general-purpose.
Esso possiede capacità di input/output, capacità di indirizzare la memoria, è in grado di manipolare dati e possiede un set di istruzioni potente.
Le relazioni tra la struttura hardware ed i termini funzionali è mostrata in figura 1-2.
Queste funzioni ed alcune loro particolarità saranno descritte nei paragrafi seguenti.

1.3.2 Capacità di input/output

Le capacità di input/output di SC/MP sono riassunte nella figura 1-3.
Per quanto riguarda il trasferimento di dati in parallelo, il microprocessore comunica mediante un bus di indirizzamento tri-state a 12 bit ed un bus dati bidirezionale tri-state ad 8 bit.
Sono inoltre disponibili due porte seriali - una per l'input ed una per l'output.
Per realizzare delle applicazioni in real-time, è possibile utilizzare un input di test, una linea di interruzione e dei flags.

Alcune linee di controllo puramente hardware presiedono all'accesso del bus da parte del microprocessore, alla priorità di accesso, al flusso dei dati ed ad un controllo generale del processore.
Le linee di controllo all'accesso del bus ed alla sua priorità possono essere utilizzate per connettere in cascata più processori, tutti aventi in comune il bus di indirizzamento e dei dati,
Se SC/MP è l'unico utilizzatore dei suddetti bus, le linee di controllo dell'accesso al bus possono essere connesse in modo tale da rendere questo accesso non interrompibile su entrambi i bus.
Le temporizzazioni dei trasferimenti di dati di input/output saranno descritti nel capitolo 2.

Comunque la sequenza di controllo di una operazione di input/output può essere riassunta come segue:

* Richiesta di accesso al bus da parte di SC/MP
* La richiesta è accolta o rifiutata; se rifiutata, la richiesta di accesso resta attiva sinchè non viene soddisfatta.
* SC/MP presenta in uscita gli indirizzi.
* Vengono presentati in uscita dei dati, oppure vengono letti dei dati.

1.3.3 Capacità di accedere alla memoria

SC/MP possiede una capacità di indirizzamento di 16 bit, conseguentemente, esso può indirizzare qualsiasi cella di memoria su 65.536.
Come si può vedere in figura 1-4, quattro bit di indirizzo sono presentati attraverso il bus dei dati, selezionando così una su 16 "pagine" di memoria, mentre sulle 12 linee degli indirizzi viene presentato l'indirizzo della cella entro la pagina.
Lo spazio di memoria può essere indifferentemente composto da ROM, PROM o RAM senza alcuna limitazione.

SC/MP utilizza cinque metodi diversi per generare un indirizzo: relativo al program counter PC, immediato, indicizzato,auto-indicizzato ed implicito. Se il campo di indirizzamento non è più di 127 bytes sopra o 128 bytes sotto l'indirizzo indicato dal program counter, si può usare il metodo di indirizzamento relativo al program counter PC. In questo modo l'indirizzo effettivo che si ottiene è la somma del contenuto del program counter e del campo spostamento (displacement) dell'istruzione (cioè del secondo byte dell'istruzione stessa). Il modo immediato usa semplicemente il secondo byte dell'istruzione come il dato da trattare; questo metodo risulta essere abbastanza veloce. in quanto non si deve eseguire un accesso in memoria addizionale per prelevare il dato necessario. Nel modo di indirizzamento indicizzato, il campo

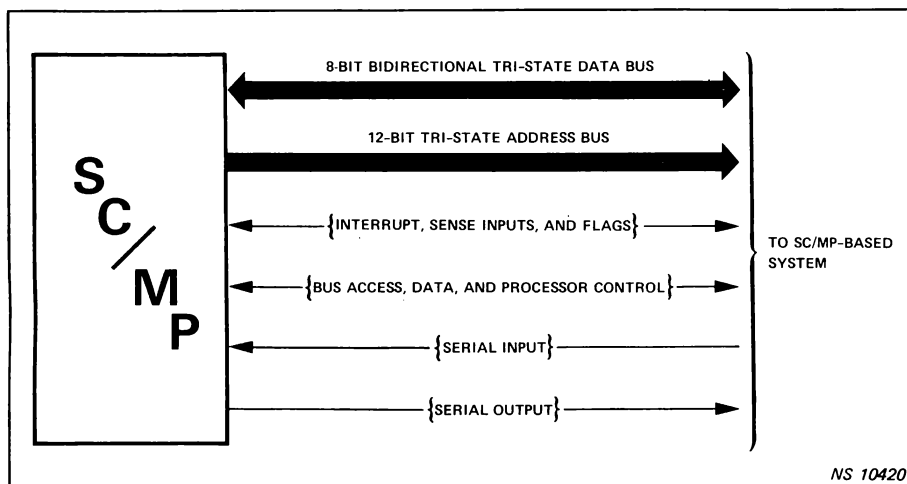



Figura 1-3. Struttura di Input/Output di SC/MP.

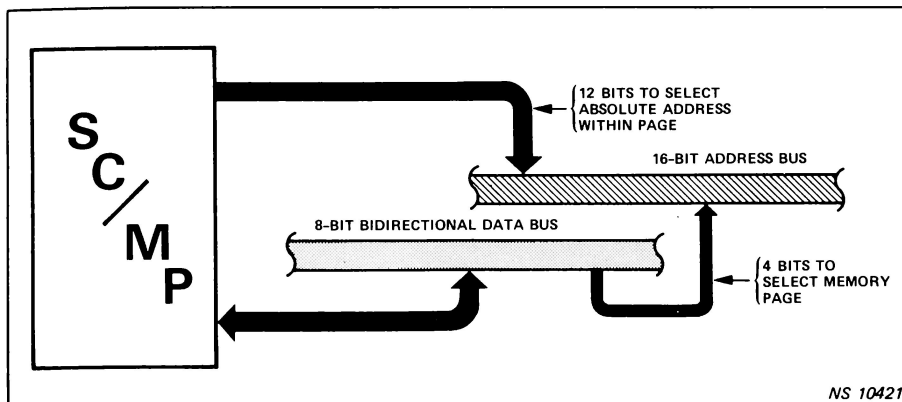



Figura 1-4. Capacità di accesso alla memoria di SC/MP.

spostamento dell'istruzione viene sommato al contenuto di uno dei registri puntatori ed il risultato è l'indirizzo del dato da trattare. L'indirizzamento autoindicizzato è simile a quello indicizzato, tranne che l'indirizzo effettivo ottenuto va a sostituire il contenuto del registro puntatore. Si veda comunque il capitolo 2 (2.4.4) per i dettagli relativi ai vari metodi di indirizzamento. Entrambi i due indirizzamenti indicizzati sono efficienti nel caso di trasferimento di blocchi di dati, lettura, o scrittura di tabelle di dati e in tutte le applicazioni simili. In ogni caso il campo di indirizzamento non deve eccedere di +127 o −128 bytes il contenuto del registro puntatore cui si fa riferimento. Il modo implicito utilizza lo stesso codice operativo per indicare gli indirizzi d'origine e di destinazione del dato. Ad esempio l'istruzione CSA (Copy Status Register to Accumulator-codice operativo = 06) definisce lo Status Register come registro d'origine e l'accumulatore come registro di destinazione.

1.3.4 Capacità di elaborazione della CPU

Le capacità di SC/MP sono innanzitutto funzione del set di istruzioni e dell'hardware che le rende possibili; questi due elementi sono schematizzati nella tabella 1-5. Il set d'istruzioni è di tipo general-purpose, dà la possibilità di utilizzare molte tecniche di programmazione ed è facile da utilizzare. D'altra parte, l'hardware è flessibile per quanto riguarda l'Input/Output, possiede un set di registri di controllo e di deposito di dati ed è semplice da interfacciare con il sistema dell'utente.

Tabella 1-5.
— *Oscillatore e generatore di clock*
Genera le temporizzazioni necessarie per il controllo di tutte le funzioni di SC/MP.
— *Logica di controllo di Input/output*
Genera tutti i segnali per lo scambio di dati con il sistema dell'utilizzatore.
— *Logica di decodifica delle istruzioni e di controllo*
Decodifica le istruzioni e provvede a tutte le funzioni di controllo interno necessarie all'esecuzione della particolare istruzione letta.
— *Registro delle istruzioni*
Memorizza il primo byte dell'istruzione durante la fase di fetch e mantiene questa informazione durante l'esecuzione dell'istruzione.

segue

Tabella 1-5

— Registro di indirizzamento

Contiene l'indirizzo della cella di memoria desiderata. Durante lo strobe degli indirizzi, esso presenta i 4 bit più significativi sul bus dei dati, mentre i 12 bit meno significativi sono presentati sul bus di indirizzamento.

— Registro dei dati di Input/Output

Durante la fase di input, questo registro di 8 bit riceve un dato dal bus dei dati e lo memorizza affinchè il processore possa poi trattare l'informazione ricevuta. Durante la fase di output il contenuto di questo registro viene semplicemente posto sul bus dei dati.

— Logica di shift, rotazione trasferimento dati

Questo circuito realizza le suddette funzioni logiche ed inoltre provvede a varie funzioni interne durante l'esecuzione dell'istruzione.

— Unità aritmetico-logica-ALU

Realizza semplicemente varie funzioni aritmetiche e logiche, quali ad esempio ADD, AND, OR, EXOR.

— Registro degli stati

E un registro ad 8 bit nel quale vengono poste le informazioni di stato relative alle operazioni aritmetiche, di controllo o di software.

— Registro extension

Questo registro a 8 bit è d'aiuto all'accumulatore per quanto riguarda l'esecuzione di operazioni aritmetiche, logiche e di trasferimento di dati. Inoltre le operazioni di Input/Output seriale sono realizzate tramite questo registro.

— Accumulatore

È un registro a 8 bit ed è per i dati la via di transito tra la memoria e i vari registri controllati da software; viene utilizzato durante l'esecuzione delle varie istruzioni aritmetico-logiche.

— Program Counter (Registro Puntatore O)

Indica l'indirizzo dell'istruzione in esecuzione, quindi esso è continuamente incrementato durante lo svolgimento del programma.

— Registro Puntatore 1

Registro a 16 bit utilizzabile come deposito temporaneo di dati o come registro di indirizzamento.

— Registro Puntatore 2

Registro a 16 bit utilizzabile come deposito temporaneo di dati o come registro di indirizzamento.

— Registro Puntatore 3

Registro a 16 bit utilizzabile come deposito temporaneo di dati o come registro di indirizzamento.

segue

Tabella 1-5

– Set di istruzioni

- LOAD
- STORE
- AND
- OR
- EXCLUSIVE-OR
- DECIMAL ADD
- ADD
- COMPLEMENT AND ADD
- LOAD IMMEDIATE
- AND IMMEDIATE
- OR IMMEDIATE
- EXCLUSIVE-OR IMMEDIATE
- ADD IMMEDIATE
- DECIMAL ADD IMMEDIATE
- COMPLEMENT AND ADD IMMEDIATE
- JUMP
- JUMP IF POSITIVE
- JUMP IF ZERO
- JUMP IF NOT ZERO
- INCREMENT AND LOAD
- DECREMENT AND LOAD
- DELAY
- LOAD ACC FROM EXT
- EXCHANGE ACC AND EXT
- AND EXTENSION
- OR EXTENSION
- EXCLUSIVE-OR EXTENSION
- ADD EXTENSION
- DECIMAL ADD EXTENSION
- COMPLEMENT AND ADD EXTENSION
- EXCHANGE POINTER LOW
- EXCHANGE POINTER HIGH
- EXCHANGE POINTER WITH PROGRAM COUNTER
- SERIAL I/0
- SHIFT RIGHT
- SHIFT RIGHT WITH LINK
- ROTATE RIGHT
- ROTATE RIGHT WITH LINK
- HALT
- CLEAR CARRY LINK
- SET CARRY LINK
- COPY STATUS TO ACC
- ENABLE INTERRUPTS
- DISABLE INTERRUPTS
- LOAD ACC FROM EXT

1.4 SC/MP E PRODOTTI DI SUPPORTO

1.4.1 Sistema di sviluppo SC/MP

La parte superiore della fig. 1-6 indica come SC/MP sia supportato da un sistema di sviluppo di tipo general-purpose di basso costo. Questo sistema grazie proprio alle sue caratteristiche, è particolarmente adatto per il progetto, la verifica ed il test del software e dell'hardware di coloro che non prevedono un utilizzo di SC/MP in grossi volumi di produzione. Il lavoro di test è facilitato da un pannello e da indicatori mediante i quali è possibile l'esame e la modifica del software in sviluppo. Inoltre il sistema di sviluppo a basso costo ha l'interfacciamento ed il firmware necessario alle periferiche (ad es. la TTY) utili allo sviluppo del software dell'utente. Questo software, una volta testato, può essere posto su nastro perforato e quindi utilizzato per produrre le memorie ROM o PROM necessarie al sistema finale dell'utente.

1.4.2 Sistema di sviluppo universale

La parte inferiore della fig. 1-6 mostra invece le caratteristiche del sistema di sviluppo cosiddetto universale. Questo sistema di sviluppo possiede delle caratteristiche software e hardware tali da permettere praticamente lo sviluppo di qualsiasi applicazione di SC/MP in tempi molto brevi e quindi con efficienza elevata. Questo sistema è tale che l'utilizzatore può caricare, assemblare e testare i programmi in sviluppo su di un calcolatore non utilizzante come CPU il microprocessore SC/MP e quindi con un set di periferiche molto vario e vasto.

1.4.3 Hardware disponibile a livello chip

Per poter estendere le possibilità d'applicazione di SC/MP sono disponibili un certo numero di componenti che verranno descritti nelle loro funzioni e nelle loro possibilità d'uso nei paragrafi seguenti.

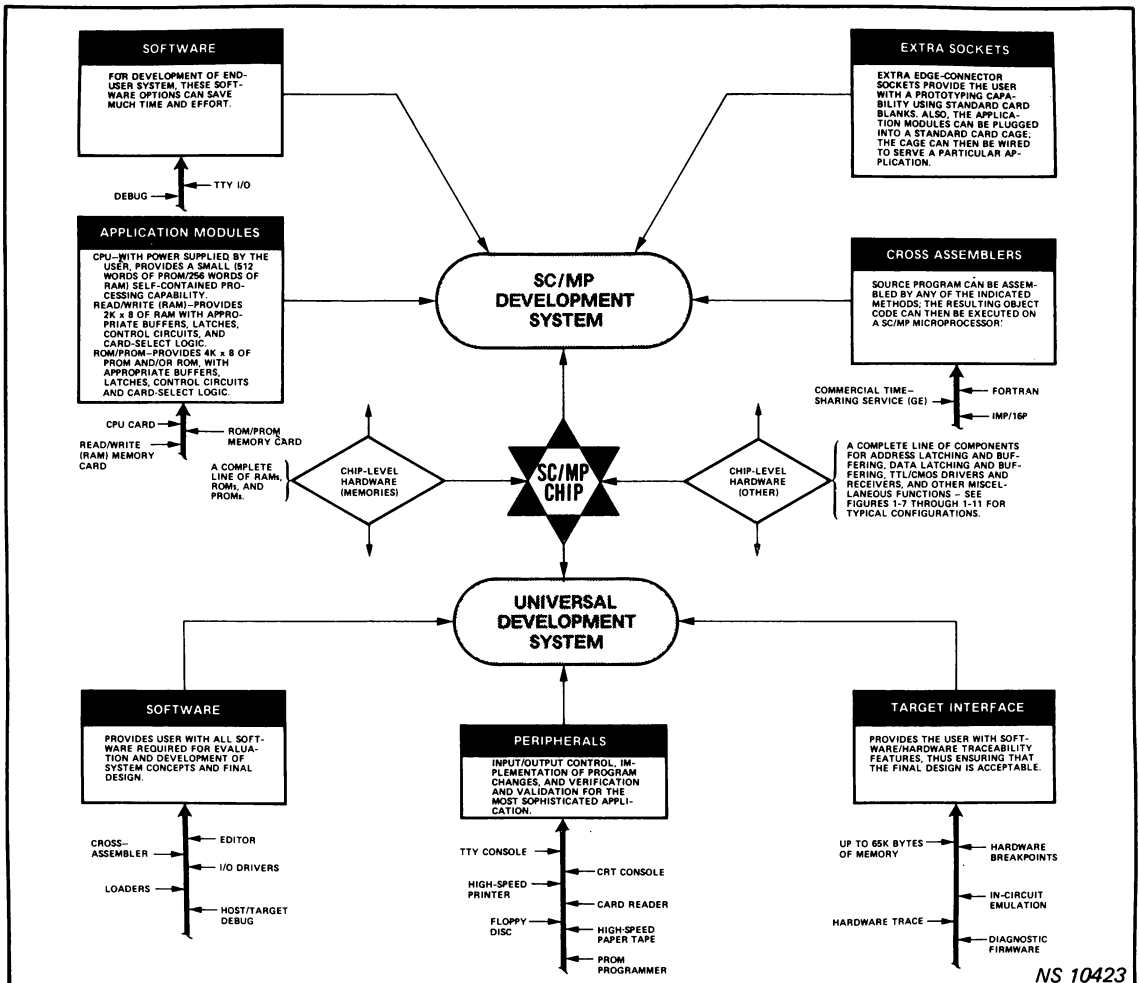



Fig. 1 - 6. SC/MP e prodotti di supporto

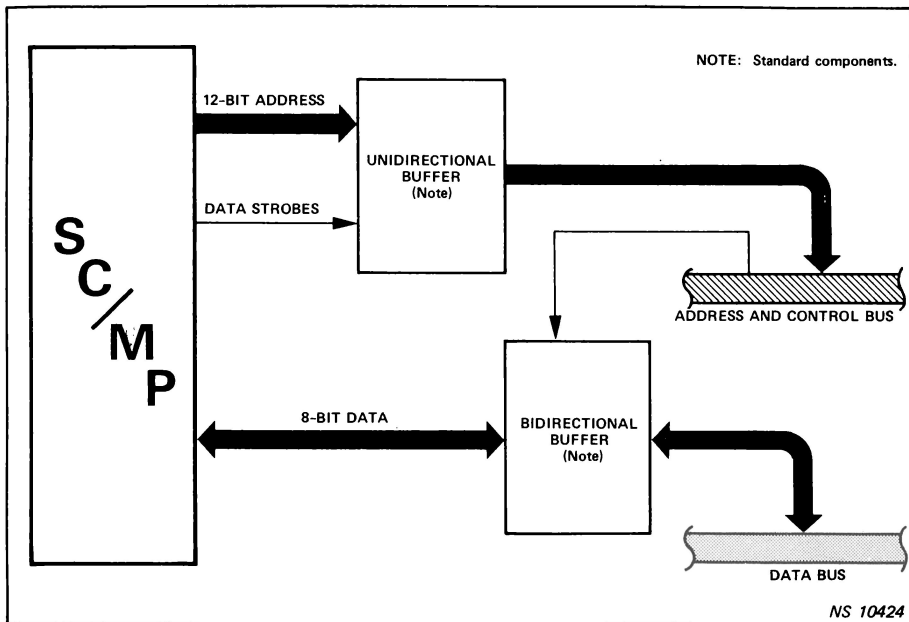
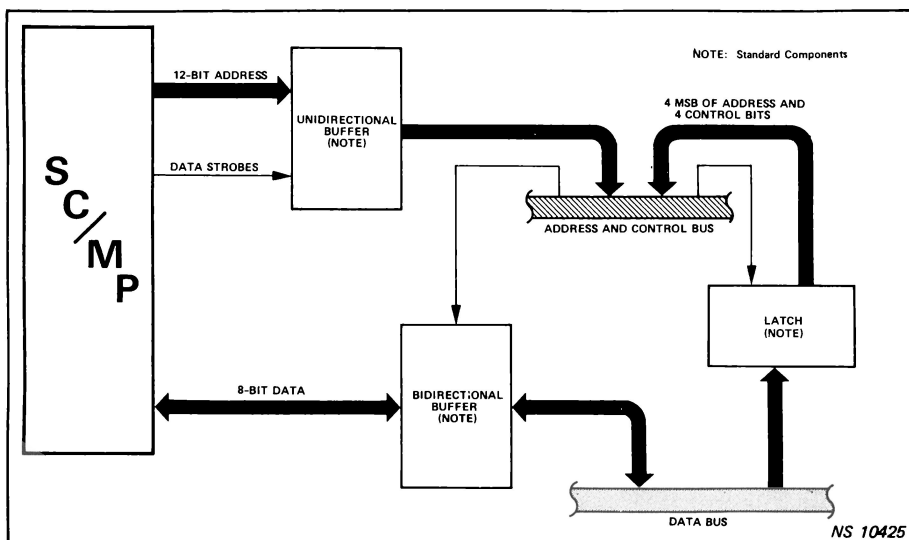



Fig. 1 - 7. Aumento del fan-out di SC/MP

1.4.3.1 Buffers

Nelle applicazioni in cui non vengono superate le caratteristiche di carico e di fan-out di SC/MP, questo microprocessore può essere utilizzato direttamente per comandare i bus degli indirizzi e dei dati. Altrimenti le linee di indirizzamento dei dati e di controllo devono essere bufferizzate come indicato in fig. 1-7. Come si può notare, il bus dei dati deve essere connesso a dei buffer di tipo bidirezionale, in questo modo i bus del sistema sono in grado di comandare un buon numero di componenti TTL, ma conservano le caratteristiche funzionali di quelli del processore.

1.4.3.2 Latches

Per aumentare le capacità di indirizzamento e di interfacciamento di SC/MP è necessario l'uso di latches. La fig. 1-8 mostra come si può facilmente

Fig, 1 - 8. Utilizzo di latches per l'espansione del sistema SC/MP

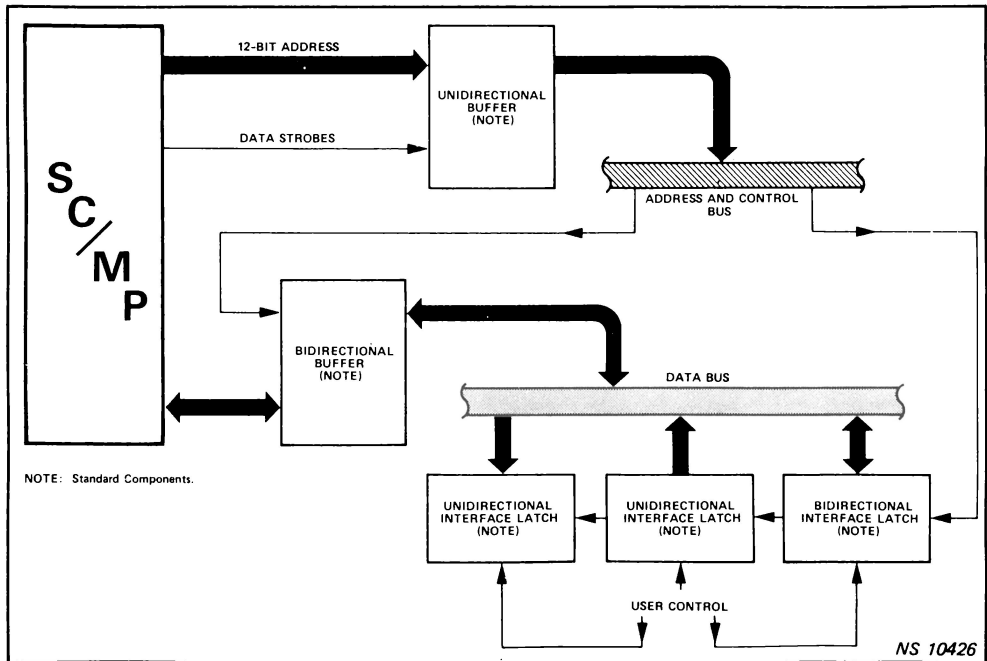



Fig. 1 - 9. Interfacciamento con SC/MP

espandere la linea di indirizzamento e di controllo di SC/MP. Scegliendo opportunamente buffers e latches, è possibile realizzare un sistema in grado di utilizzare sino a 65K bytes di memoria con timing e logiche di controllo semplici.

La fig. 1-9 indica come si possono utilizzare latches per l'interfacciamento. I chip di interfacciamento possono essere utilizzati come porta Input/Output bidirezionali, oppure come porta di solo Input o solo Output. Evidentemente il colloquio con il sistema dell'utente

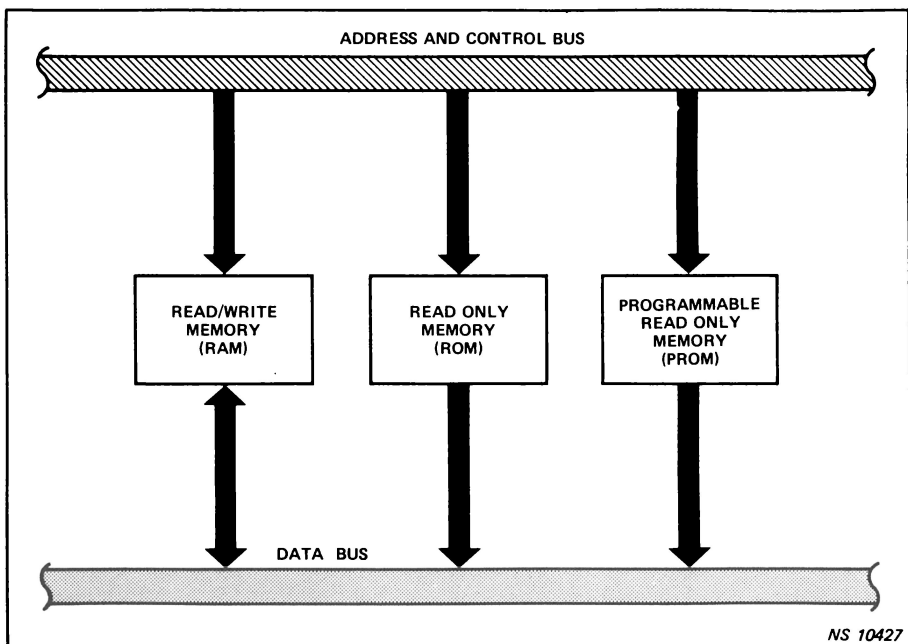



Fig. 1 - 10. Collegamento di memorie con SC/MP

sarà di tipo statico nel caso di utilizzo di una porta di solo Input ed una di solo Output, mentre sarà di tipo dinamico nel caso di utilizzo di porta di Input/Output bidirezionale. Inoltre, in quest'ultimo caso,il latch deve essere di tipo tri-state.

1.4.3.3 Memorie

SC/MP è compatibile in modo diretto con molte memorie standard, quali RAM (Random Access Memories), ROM (Read Only Memories) e PROM (Programmable Read Only Memories). Come mostrato in figura 1-10, l'indirizzamento e l'accesso alle memorie suddette è controllato dal processore SC/MP che genera tutti i segnali di controllo necessari allo scopo. Questi segnali di controllo permettono di indirizzare facilmente i vari chip di memoria e di realizzare il trasferimento dei dati. I circuiti di buffer sono necessari solamente in funzione del carico totale esistente sui bus del sistema.

Infine la fig. 1-11 mostra un sistema completo utilizzante SC/MP come CPU e componenti standard per le funzioni di interfacciamento e bufferizzazione.

1.5 SCHEDE APPLICATIVE DI SC/MP

Le seguenti schede di applicazione aiutano il progetto di un sistema che utilizza il microprocessore SC/MP:

- SC/MP CPU Application Module
- SC/MP RAM Application Module
- SC/MP ROM/PROM Application Module

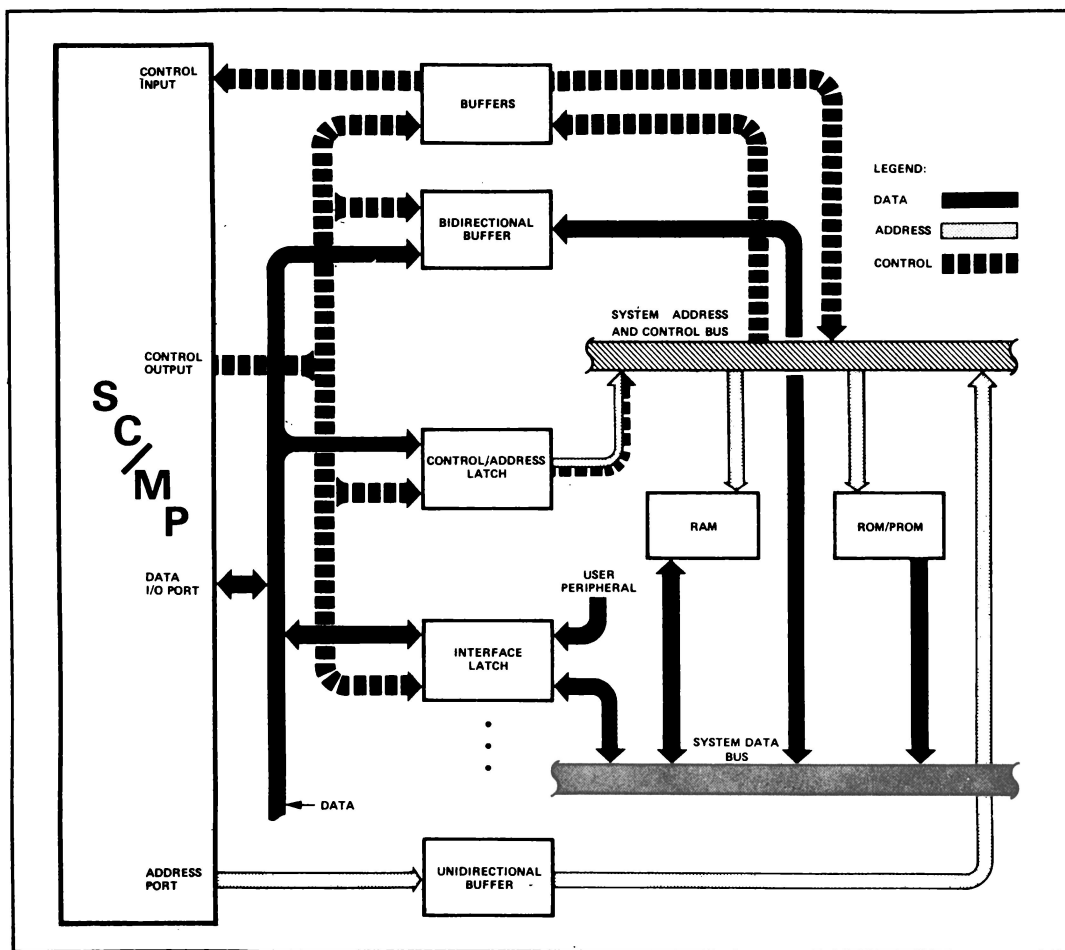



Fig. 1 - 11 Sistema tipico di connessione di SC/MP in un sistema completo

Ciascuna scheda ha il formato di 100x160mm e può essere posta in porta schede di tipo standard. Esse possono essere facilmente collegate l'una con l'altra formando così il sistema desiderato dall'utilizzatore, diminuendone d'altra parte il tempo ed il costo di sviluppo.

1.6 IL SOFTWARE DI SC/MP

L'importanza del software di supporto è veramente molto grande e lo sviluppo di una particolare applicazione sarà efficiente solamente se il progettista lo saprà comprendere ed utilizzare appieno. Attualmente SC/MP possiede, oltre ad un *assembler residente* (che opera cioè sul sistema di sviluppo a basso costo), cross assembler, caricatore, programma di debug, routines di Input/Output.

1.6.1 Cross Assembler (per IMP-16)

Il cross assembler accetta programmi simbolici sia da tastiera che da nastro o da schede. Ogni programma genera un modulo oggetto (LM) su nastro perforato ed un listing del programma. L'assemblatore richiede tre letture del programma sorgente.

Fig. 1 - 12. Cross Assemblers di SC/MP Loro Flow operativo

1.6.2 Cross Assembler (Fortran)

Questo programma, scritto in Fortran IV, assembla un programma sorgente affinchè poi questo programma sia eseguibile da SC/MP. Questo programma, in due passi, genera un modulo oggetto e un listing del programma; è disponibile sul servizio timesharing General Electric con il nome SAS $$$.

1.6.3 Caricatore Assoluto

Questo programma carica uno o più programmi in aree prefissate di memoria. La zona precisa di memoria occupata da ogni programma caricato deve essere determinata dall'utilizzatore prima dell'assemblaggio Inoltre, ogni eventuale legame diretto tra due programmi o con una zona comune di memoria, deve essere effettuato nel momento dell'assemblaggio mediante l'assegnamento di etichette uguali che si riferiscono a medesimi indirizzi in memoria.

1.6.4 Routines di gestione teletype

Queste routines possono essere utilizzate per ricevere e trasmettere informazioni da e per una teletype (TTY). Inoltre, per quanto riguarda le routines di ricezione è possibile ricevere un carattere da tastiera senza eco, oppure eseguendo l'eco del carattere ricevuto alla stampante della TTY.

1.6.5 Programma di Debug di SC/MP

Il programma di Debug controlla semplicemente l'esecuzione di un programma utente durante il suo test Esso cioè provvede a fornire le seguenti possibilità di test:
- Stampa del contenuto di una zona di memoria in codice esadecimale
- Modifica del contenuto di una zona di memoria
- Visualizzazione e modifica dei registri della CPU
- Inserzione di punti automatici di halt
- Iniziare l'esecuzione del programma utente da qualsiasi punto

Capitolo 2°

Fig. 2 - 1. Schema a blocchi funzionale dello SC/MP e configurazione dei pin di uscita

2.1. DESCRIZIONE FUNZIONALE

La fig. 2-1 è uno schema a blocchi funzionale del chip SC/MP. I segnali di Input/Output mostrati in detta figura saranno descritti nella tabella 2-1, mentre le varie unità funzionali saranno presentate in dettaglio in questo capitolo. È da precisare comunque che tutte queste descrizioni sono da vedersi da un punto di vista applicativo. Il set di istruzioni di SC/MP viene presentato dettagliatamente nell'appendice A.

Nota.
1. In questo manuale viene usata come convenzione la logica positiva. Un "1" logico, o segnale alto, corrisponde al livello di tensione piú positivo, mentre uno "0" logico, o segnale basso, corrisponde al livello di tensione più negativo. Tutti i nomi di segnali che iniziano con "N" e sono seguiti da un asterisco (*) indicano segnali la cui azione e valida quando sono allo stato logico "O".
2. I bits sono numerati da 00 a 15, da destra a sinistra, con il bit 00 che indica il bit meno significativo.
3. La X' che precede un numero indica che questo ultimo è un valore espresso secondo il sistema di numarazione esadecimale.

2.2 ALIMENTAZIONE E TIMING

Come mostrato in fig. 2-2, il chip SC/MP può essere alimentato da una singola alimentazione, cioè con —12 Volt rispetto al pin Vss. Se si deve però realizzare un interfacciamento diretto con logiche TTL o CMOS operanti a 5 Volt, è necessario alimentare SC/MP come indicato in fig. 2-2a..Qualora però siano presenti anche componenti quali PROM, che necessitano cioè di 17 Volt di alimentazione, è necessario alimentare il sistema secondo la fig. 2-2b..

Tutte le temporizzazioni necessarie sono fornite da un oscillatore e da un generatore di clock interno. Se non si ha necessità di timing precisi, si può collegare semplicemente un condensatore ai pin X_1 e X_2 di SC/MP. In quelle applicazioni in cui il timing deve essere accurato, è sufficiente utlizzare un quarzo connettendolo ai pin X_1 e X_2.

Nota.
Il quarzo da utilizzare deve essere posto in un contenitore stagno e può essere acquistato, indicativamente, presso i seguenti produttori:

- Betron S.A. Livorno, Italia
- Quartz et Electrique, France
- Cepe, France
- R.T.C., France
- Fischer, Germany
- Siemens, Germany
- AEG-Telefunken, Germany
- ITT, Germany
- Sylvania N.V.-GTE, Belgium
- Valvo, Germany
- Cathodeon Crystals, Cambrige, England

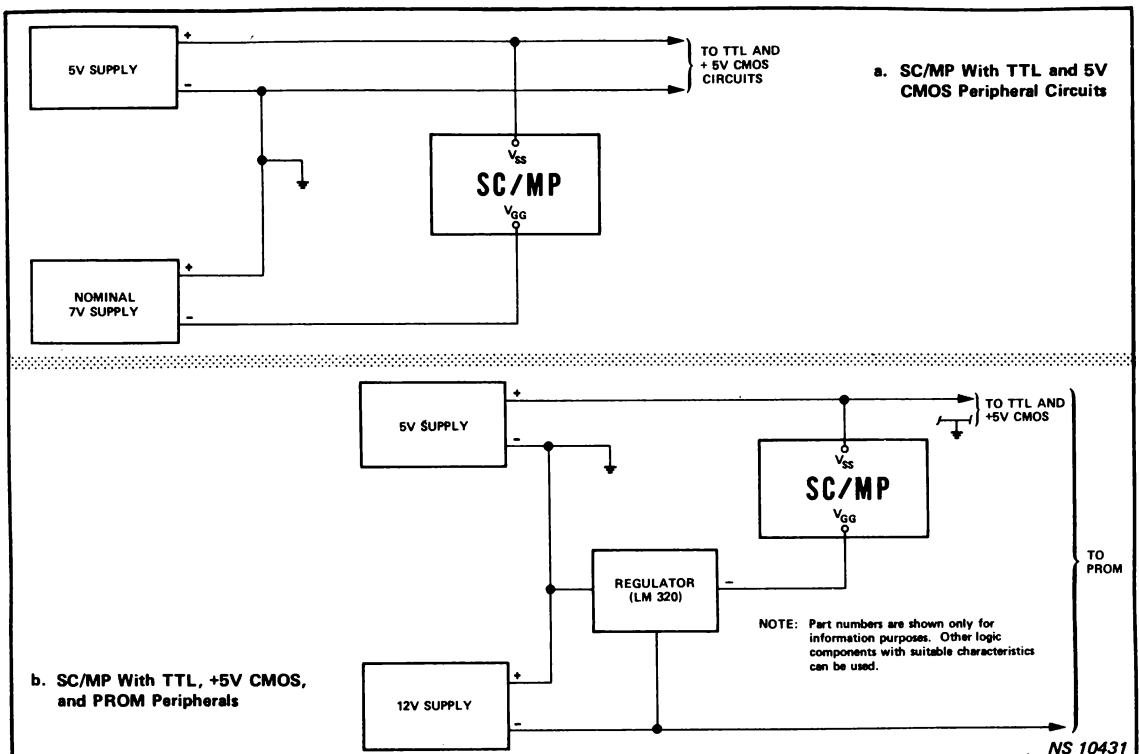



Fig. 2 - 2 Metodi possibili di alimentazione di SC/MP

2-3 LOGICA DI INPUT/OUTPUT

I dati e gli indirizzi di SC/MP sono presenti su due bus separati: un bus bidirezionale di 8 bit per i dati ed un bus di indirizzamento a 12 bit. Tramite questi bus si possono realizzare trasferimenti di dati in parallelo e quindi velocemente. Per un trasferimento di dati lento, si può utilizzare la logica di Input/Output seriale.

Tabella 2 - 1 Descrizione dei segnali di Input/Output

Codice mnemonico del segnale	Nome funzionale	Descrizione
X_1/X_2		Connettere una capacità od un quarzo per realizzare il timing di SC/MP.
Vss		Tensione di alimentazione positiva.
Vgg		Tensione di alimentazione negativa.
NRST (Input)	Reset	Quando è basso, le operazioni in corso vengono interrotte, quando ritorna da basso ad alto, resetta tutti i registri interni di SC/MP ed il processore legge la prima istruzione dalla cella X'0001.
CONT (Input)	Continue	Quando è alto, l'istruzione viene letta dalla locazione indicata dal program counter e quindi eseguita. Quando è basso, il processore si blocca al termine dell'esecuzione in corso. Agendo su questo ingresso è quindi possibile realizzare l'esecuzione di un programma in modo di singola istruzione.
BREQ (I/O)	Bus Request	A seconda della configurazione del sistema, questa linea viene usata come segnale di bus-request e/o bus-busy. La linea BREQ può essere connessa in "wired-AND" e necessita di una resistenza di carico a VGG o massa.
ENIN (Input)	Enable In	Quando è alta, permette al processore l'accesso al bus di Input/Output.
ENOUT (Output)	Enable Out	Quando è alta, indica che ENIN è alto e che la CPU non sta occupando il bus. Quando è bassa, indica che la CPU sta accedendo al bus, oppure che ENIN è basso.
NADS (Output)	Address Strobe	Quando è basso, i 4 bit di stato ed i 4 bit significativi di indirizzo sono presenti sul bus dei dati.
NRDS (Output)	Read Strobe	Quando è basso, i dati in ingresso al processore devono essere validi sul bus dei dati. Quando il processore non ha accesso al bus, questa linea è in condizioni di alta impedenza (tri-state).
NWDS (Output)	Write Strobe	Quando è basso, i dati in uscita del processore sono validi sul bus dei dati. Quando il processore non ha accesso al bus, questa linea è in condizioni di alta impedenza (tri-state).
NHOLD (Input)	Hold o Extended	Quando è bassa, il ciclo di Input/Output del processore viene allungato sinché questa linea non ritorna alta. Poiché in questo modo si allungano anche gli impulsi di NWDS e NRDS, questa linea permette di interfacciare SC/MP con logiche anche molto lente.
SENSE A (Input) SENSE B (Input)	Sense Input- Interrupt Request Sense Input	Queste due linee sono testate dal processore leggendo i bit 4 e 5 del registro di stato; sono lette in modo sincrono. Sense A inoltre opera come ingresso di richiesta di interruzione se abilitato da software.

segue

Tabella 2 - 1

Codice mnemonico del segnale	Nome funzionale	Descrizione
SIN (Input)	Serial Input al registro E.	Mediante controllo software, il dato presente su questa linea viene shiftato a destra nel registro Extention mediante l'istruzione SIO.
SOUT (Output)	Serial Output del registro E.	Sotto controllo software, il contenuto del registro Extention viene shiftato a destra su questa linea. Questa operazione si attua eseguendo l'istruzione SIO. Il dato presentato sulla linea SOUT viene memorizzato, quindi resta stabile anche se il contenuto del registro Extention viene modificato.
FLAG, 0, FLAG 1, FLAG 2	Flag Output	Questi tre flag corrispondono rispettivamente ai bit 0, 1, 2, del registro di stato.
AD 00 - AD 11	Bit di indirizzo dal bit 00 al bit 11	Sono 12 linee di indirizzamento tri-state. I dati su queste linee sono validi quando viene generato NADS. Questa informazione resta stabile sino al fronte di salita di NRDS o di NWDS. Le linee AD00-AD11 sono in tri-state (alta impedenza) quando il processore non accede al bus.

Codice mnemonico del segnale	Output quando è presente NADS			Input con NRDS	Output con NWDS
	Codice	Funzione	Descrizione		
DB 0	AD 12	Indirizzo bit 12		↑	↑
DB 1	AD 13	Indirizzo bit 13	Quattro bit più significativi di indirizzo		
DB 2	AD 14	Indirizzo bit 14			
DB 3	AD 15	Indirizzo bit 15		Dati in ingresso	Dati in uscita
DB 4	RFLG	Read Flag	Quando è alto, sta per iniziare un ciclo di Input		
DB 5	IFLAG	Instruction	Quando è alto, inizia la lettura del primo byte di un'istruzione		
DB 6	DFLAG	Delay Flag	Quando è alto, sta per iniziare un'istruzione di DLY, cioè il processore sta leggendo il secondo byte dell'istruzione di delay		
DB 7	HFLAG	Halt Flag	Quando è alto, si sta eseguendo un'istruzione di HALT.	↓	↓

Nota: Le linee DB0 — DB7 sono nello stato di alta impedenza quando SC/MP non accede al bus di Input/Output.

2-3-1 Accesso ai bus

Prima di operare un trasferimento di dati da o per la memoria e periferiche, SC/MP deve accedere ai bus del sistema. Questo accesso ai bus è controllato in modo semplice, ma efficace, come indicato in fig. 2-3. L'accesso al bus è controllato mediante tre segnali; bus request (BREQ), enable input (ENIN), ed enable output (ENOUT). Nel caso di sistemi molto semplici, BREQ e ENOUT possono essere non utilizzati, mentre ENIN deve essere sempre abilitato.
In una soluzione come questa, l'accesso ai bus è sempre controllato da SC/MP ed i trasferimenti di dati da una periferica all'altra devono sempre avvenire tramite il microprocessore. In quei sistemi che utilizzano periferiche con velocità di trasferimento dei dati molto alta, può essere utile realizzare il metodo di DMA (Direct Memory Access) per seguire i trasferimenti di dati tra periferiche e memoria. Utilizzando questa tecnica, SC/MP non interviene direttamente nel trasferimento dei dati se non per alcune operazioni di controllo. Una configurazione tipica di DMA è mostrata in fig. 2-4 con i relativi timing. Nella fig. 2-4a., l'accesso al bus è regolato da un circuito esterno al processore, cioè dal DMA Controller. In questo caso, SC/MP richiede l'accesso al bus ponendo la linea BREQ alta. Questo segnale avvisa la logica di controllo di DMA e se il bus è libero, viene posta alta la linea di ENIN, abilitando il processore ad occupare i bus. Quando il processore ha terminato il proprio ciclo di Input/Output, esso pone bassa la linea di BREQ, lasciando il controllo dei bus al DMA Controller.
Anche in questo caso, SC/MP richiede l'accesso al bus ponendo la linea BREQ alta. Questo segnale avvisa la logica di controllo di DMA e se il bus è libero, viene posta alta la linea di ENIN, abilitando il processore ad occupare i bus. Quando il processore ha terminato il proprio ciclo di Input/Output, esso pone bassa la linea di BREQ, lasciando il controll dei bus al DMA Controller.
La fig. 2-4 mostra un sistema composto da pi processori SC/MP, in cui l'accesso al bus è contro lato dalla logica interna dei processori stessi. Le fun zioni di accesso al bus e le priorità sono regola mediante tre segnali di controllo:
BREQ - ENIN - ENOUT.
La linea di BREQ dei vari processori è connessa i modo da realizzare la funzione wire-AND. A quest stessa linea è connesso l'ENIN del primo processore In questo modo se il primo processore richiede l'ac cesso al bus, esso ne ha automaticamente garanti l'accesso. Se il primo SC/MP controlla i bus, ENOUT è basso e conseguentemente tutta la catena deg altri processori è bloccata. Se per caso sia lo SC/M numero 2 che lo SC/MP numero 1 iniziano una richie sta di accesso al bus (BREQ2 e BREQN setta alti), ma il processore numero 1 sta eseguendo un operazione di Input/Output, si avrà la seguente se quenza di operazioni. ENOUT 1 è basso sinchè SC/M numero 1 non ha terminato, poi diviene alto. A quest punto, essendo ENOUT 1 alto, anche ENIN 2 è alt e di conseguenza lo SC/MP numero 2 ottiene il con trollo del bus. Riassumendo si può notare come, i una struttura così impostata, non si hanno problem di accesso ai bus nemmeno quando tutti i processo desiderano accedervi simultaneamente.
La catena di processori è abilitata secondo una scal di priorità: SC/MP numero 1 per primo, SC/MP nume ro 2 per secondo, SC/MP numero 3 per terzo e cos via. Qualsiasi processore SC/MP disegnato in fig. 2-4 potrebbe essere sostituito da una logica di DMA purchè essa sia in grado di gestire BREQ, ENIN e ENOUT esattamente nello stesso modo. In ogni caso generalmente non vengono connessi in questo mod più di tre SC/MP. Qualora sia necessario utilizzare i un sistema un numero maggiore di processori, è op portuno realizzare un Priority Controller esterno.

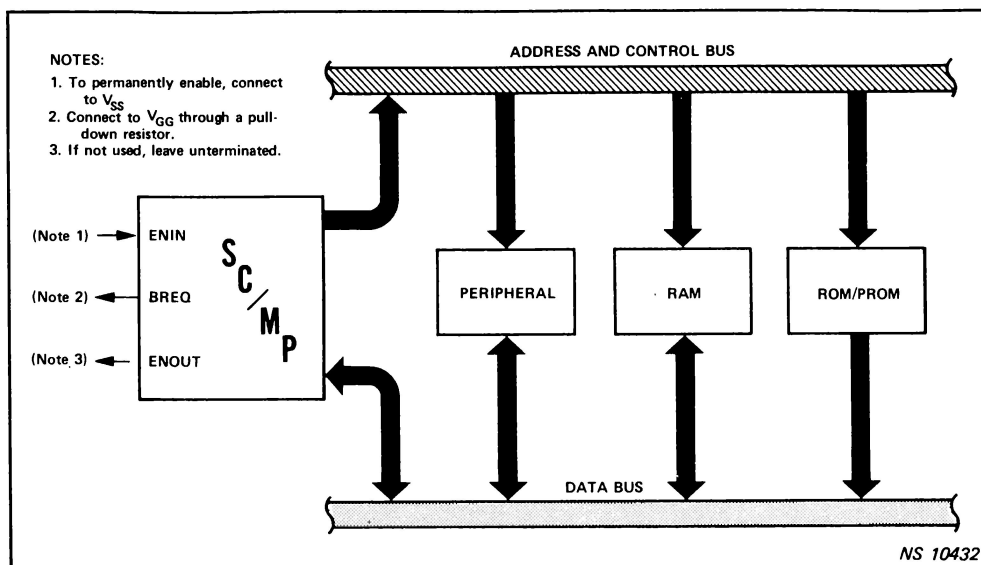



Fig. 2 - 3. Segnali di controllo per l'accesso ai bus

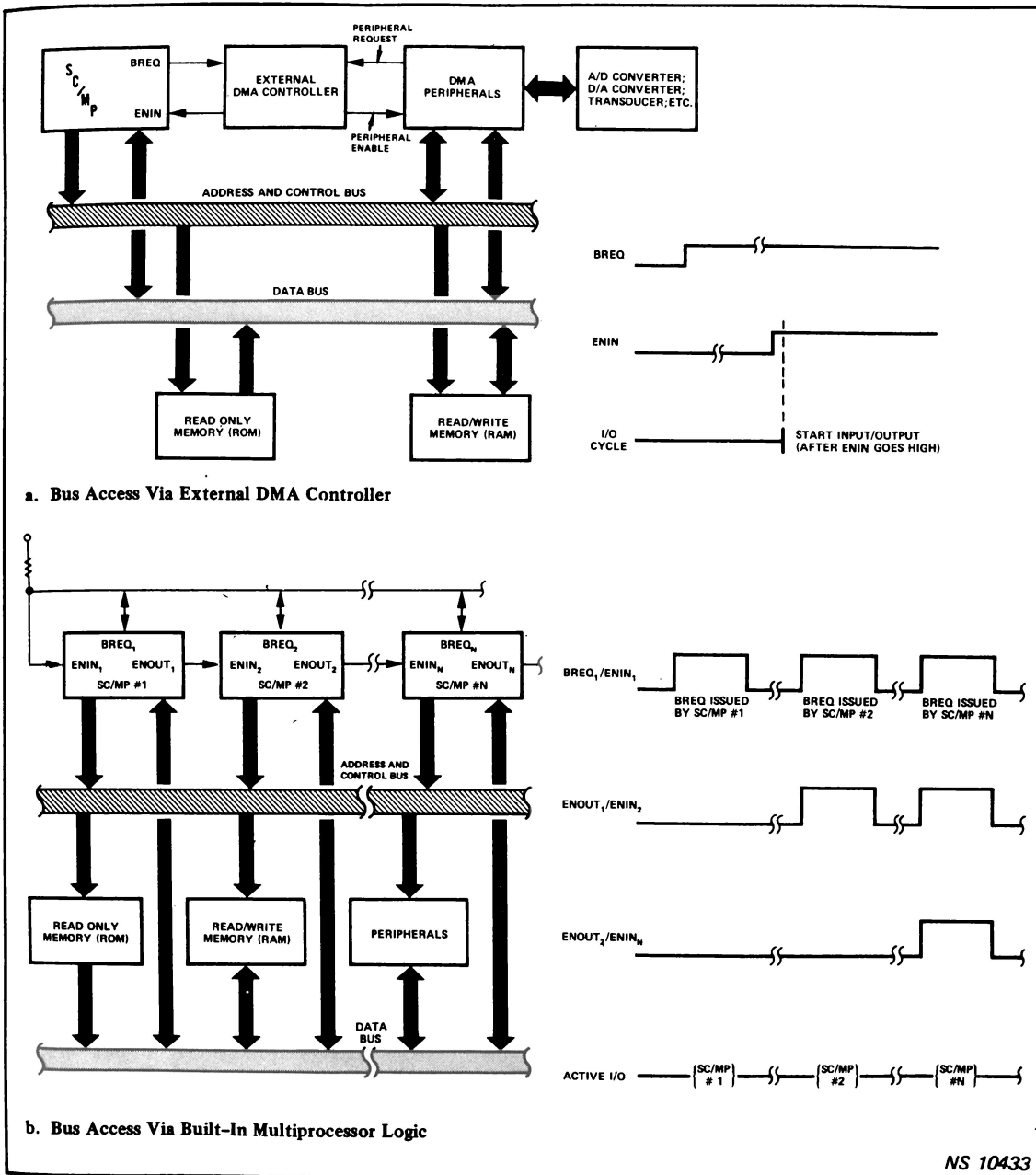



Fig. 2 -4. Struttura di un sistema multiprocessors

2.3.2 Ciclo di Input/Output

Una volta ottenuto il controllo dei bus, SC/MP inizia un ciclo di Input o di Output. Fondamentalmente un ciclo di Input consiste nella lettura di una cella di memoria (o di una periferica), mentre un ciclo di Ouput è un'operazione di scrittura in una cella di memoria (o in una periferica), La fig. 2-5 mostra sia un ciclo di scrittura che un ciclo di lettura. Questi due cicli sono stati schematizzati e non sono presenti le varie temporizzazioni, ma sono sufficientement dettagliati per comprenderne la logica.
Accettata la richiesta di accesso ai bus, BREQ e ENI sono alti; contemporaneamente il processore presen ta gli indirizzi da AD00 a AD11, indirizzando quin direttamente una cella su 4096.
Questi indirizzi sono memorizzati nel chip stesso d SC/MP, quindi restano stabili per tutta la durata de ciclo di Input/Output.

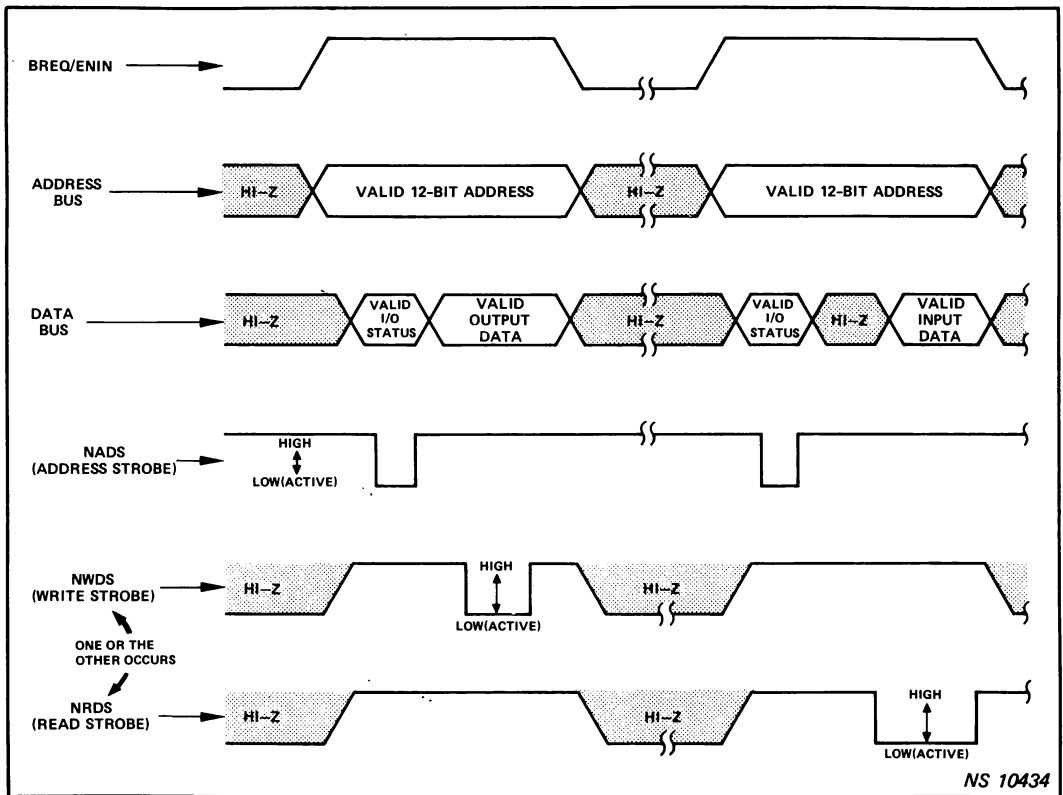



Fig. 2 - 5. Tipica sequenza di Input/Output di SC/MP

La fig. 2-6 mostra l'informazione presente sul bus dei dati quando NADS è attivo. In questa fase, vengono presentati in uscita i 4 bit di indirizzo più significativi (AD12 — AD15) ad indicare una delle possibili 16 "pagine" da 4096 celle l'una. Gli altri quattro bit sono segnalazioni (flag) di stato. I quattro bit di selezione della pagina, combinati con i dodici bit statici di indirizzamento, danno la possibilità di indirizzare 65.536 byte di memoria; tuttavia per essere utilizzati necessitano di un latch esterno. A scopo di controllo, anche il flag di stato possono essere memorizzati esternamente mediante latches. La fig. 2-7 illustra l'utilizzo di uno di questi flag, l'H-flag (bit 7 di stato) e l'input di Continue allo scopo di bloccare il processore quando questo esegue l'istruzione di HALT Quando viene premuto il pulsante S1 (viene cioè portato momentaneamente nella condizione NO) si genera un impulso che setta la linea di CONT alta Finchè DB7 non è alto in coincidenza con NADS, la linea di CLR è alta ed il processore continua ad operare. Quando però il flag di Halt viene posto alto (DB7 alto in presenza di NADS), l'ingresso di CONT viene settato basso da Q ed il processore interrompe l'esecuzione del programma.

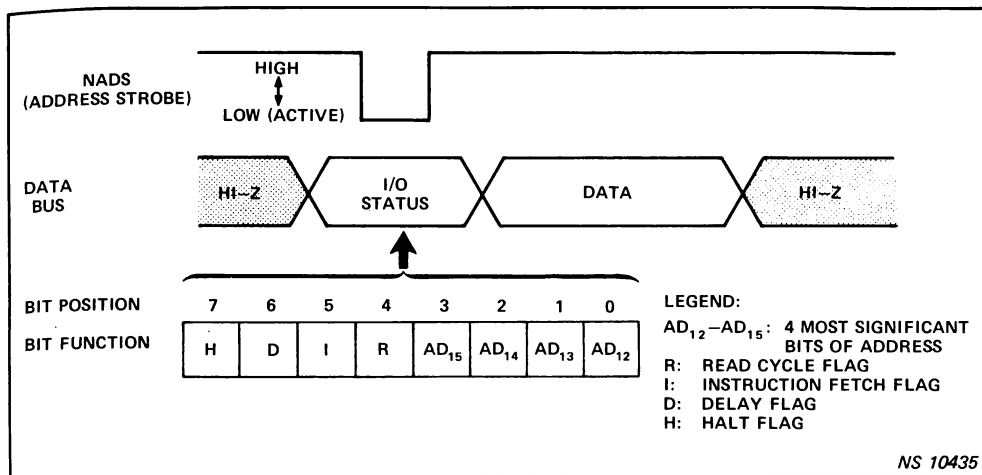



Fig. 2 - 6. Bus dei dati in presenza di NADS

Partendo dal principio di fig. 2-7 è possibile realizzare funzioni di controllo di SC/MP più complesse, quali ad esempio quelle mostrate in fig. 2-8. In questo caso, infatti, si possono realizzare le operazioni di singola istruzione e di singolo ciclo. Il commutatore S1 viene posto nel modo di operare desiderato, mentre il pulsante S2 svolge la funzione di start. Alla pressione di S2, le linee CONt e NHOLD vengono poste alte e quindi il proccessore è in grado di operare . Quando viene generato NADS, a seconda della posizione di S1, viene posta bassa la linea di CONT o di NHOLD. Se viene posta a "0" la linea di Cont il processore esegue una singola istruzione, mrbtr se viene posto a "0" NHOLD., il processore esegue un solo ciclo.

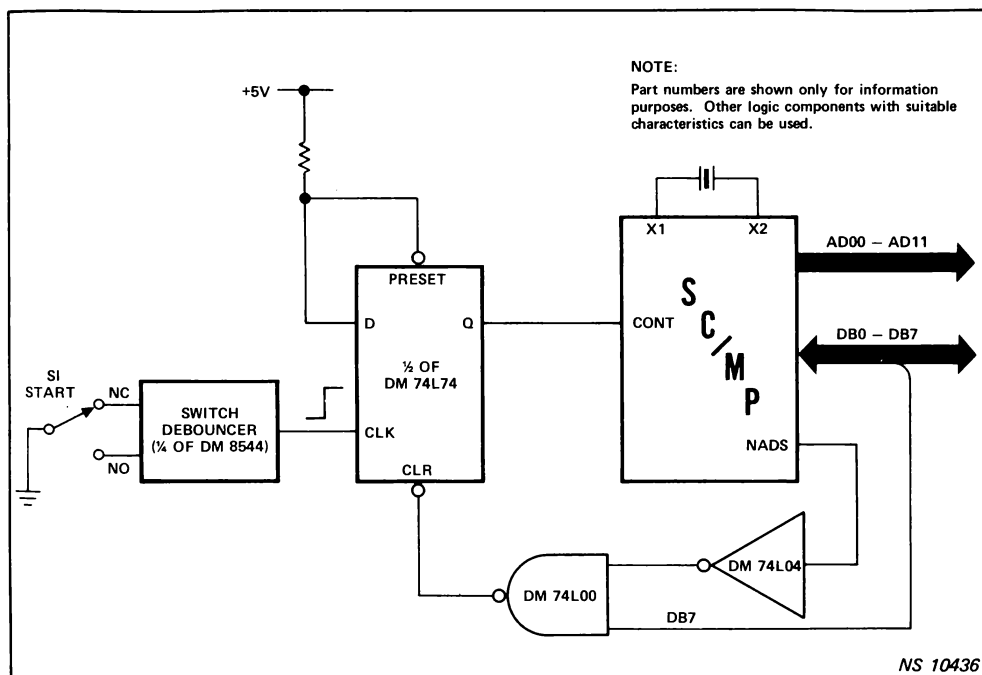



Fig. 2 - 7. Utilizzo del flag di "Halt" per generare un Halt da programma

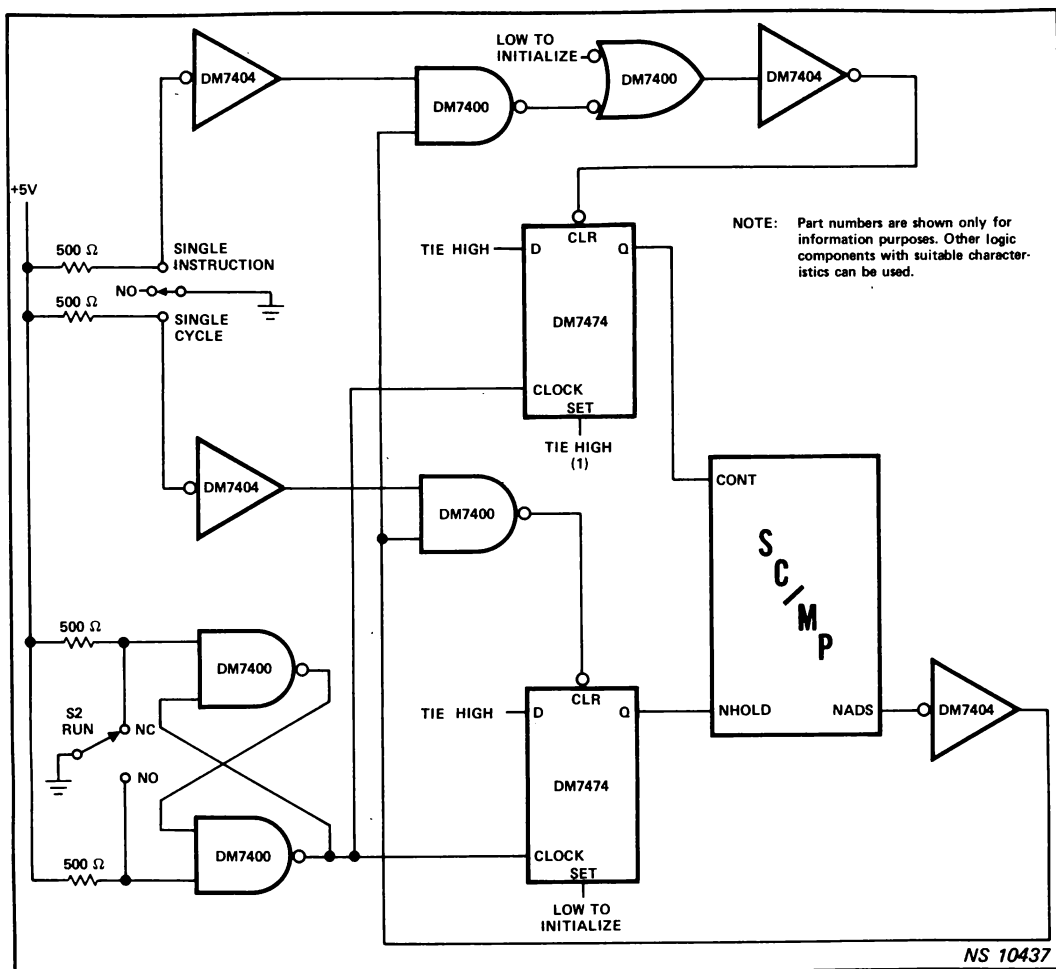



Fig. 2 -8. Circuito necessario al funzionamento di SC/MP in singola istruzione o singolo ciclo

Una volta generato l'impulso di strobe degli indirizzi, il processore può iniziare un ciclo di Input (lettura) od un ciclo di Output (scrittura). Come si può notare in fig. 2-5, le operazioni di lettura o di scrittura sono sincronizzate dagli strobe di lettura (NRDS) o di scrittura (NWDS). Quando lo strobe di lettura è basso, i dati presenti sul bus vengono letti dal processore; quando invece lo strobe di scrittura è basso, i dati presentati dal processore sul bus sono stabili.

I trasferimenti di dati da e per SC/MP non sono sincronizzati a nessun timing particolare, conseguentemente il collegamento del processore con memorie o periferiche (con tempo di accesso di 1 microsecondo o meno) è semplice. La sequenza di Input/Output è semplice. La sequenza di Input/Output è, riassumendo, la seguente:

— SC/MP richiede l'accesso al bus
— L'accesso è garantito
— Viene presentato l'indirizzo
— I dati sono scritti o letti

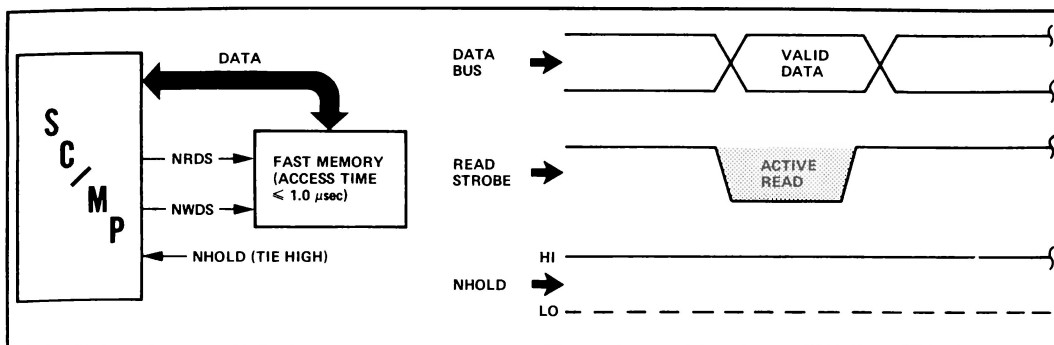
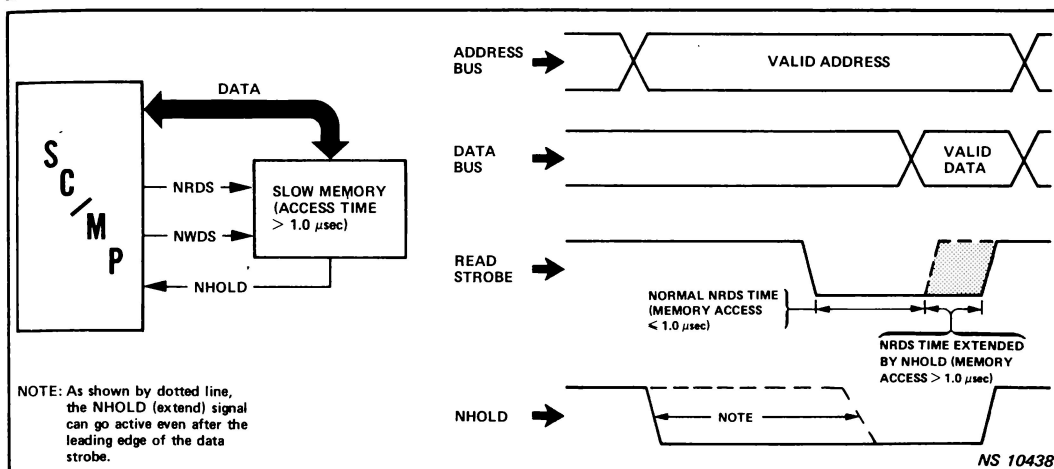
Nel caso di un ciclo di Input, i dati devono essere presentati al processore prima della transizione da basso ad alto dello strobe di NRDS. Qualora si connettano al processore SC/MP memorie di tipo lento quindi non in grado di presentare dei dati validi prima della fine dello strobe di lettura, è necessario agire sulla linea di NHOLD. Come mostrato in fig. 2-9b, lo strobe di lettura (o quello di scrittura) possono essere allungati in funzione del tempo di accesso della memoria. In questo modo la velocità di elaborazione di SC/MP resta inalterata e solamente i cicli di Input/Output saranno più lenti.

a. Memoria ad accesso veloce

b. Memoria ad accesso lento

Fig. 2 - 9. Estensione del ciclo di Input/Output di SC/MP

Fig. 2 - 10. Utilzzo di SC/MP con o senza buffers

2.3.3 Buffering dei bus di SC/MP

Se si applica SC/MP in sistemi che richiedono una piccola quantità di memoria, quale quello mostrato in fig. 2-10a., non è necessario alcun potenziamento dei bus dei dati e degli indirizzi. Qualora però il sistema richieda una certa quantità di periferiche e di memorie, può essere necessario fornire ai bus di SC/MP un elevato fan-out. In questo caso è necessario realizzare quanto indicato dalla fig. 2-10b.

2.3.4 Trasferimento seriale di dati

Se il processore SC/MP non deve ricevere od inviare dati a periferiche particolarmente veloci, il suddetto scambio può avvenire in modo seriale. La fig. 2-11 mostra appunto come si può utilizzare la capacità di SC/MP.

Innanzittutto vengono utilizzate per questo tipo di trasferimento dati le porte SIN e SOUT di SC/MP che sono connesse direttamente al registro extension.

In questo caso i flag 0,1 e 2 vengono utilizzati per il controllo dei due shift register. Inoltre le linee SIN e SOUT possono essere espanse, realizzando così più linee di Input e di Output seriale mediante dei multiplexer.

2.3.5 Flag e linee di sense

I bit 0,1,2,4 e 5 dello status register costituiscono tre flag e due Input di sense, secondo la seguente tabella:

Status Register	*Descrizione*
0	Flag 0 (Output)
1	Flag 1 (Output)
2	Flag 2 (Output)
4	Sense A (Input)
5	Sense B (Input)

Queste linee sono disponibili all'utente, quindi possono essere utilizzate sia per indicazioni di stato che per funzioni di controllo Hardware/Software.

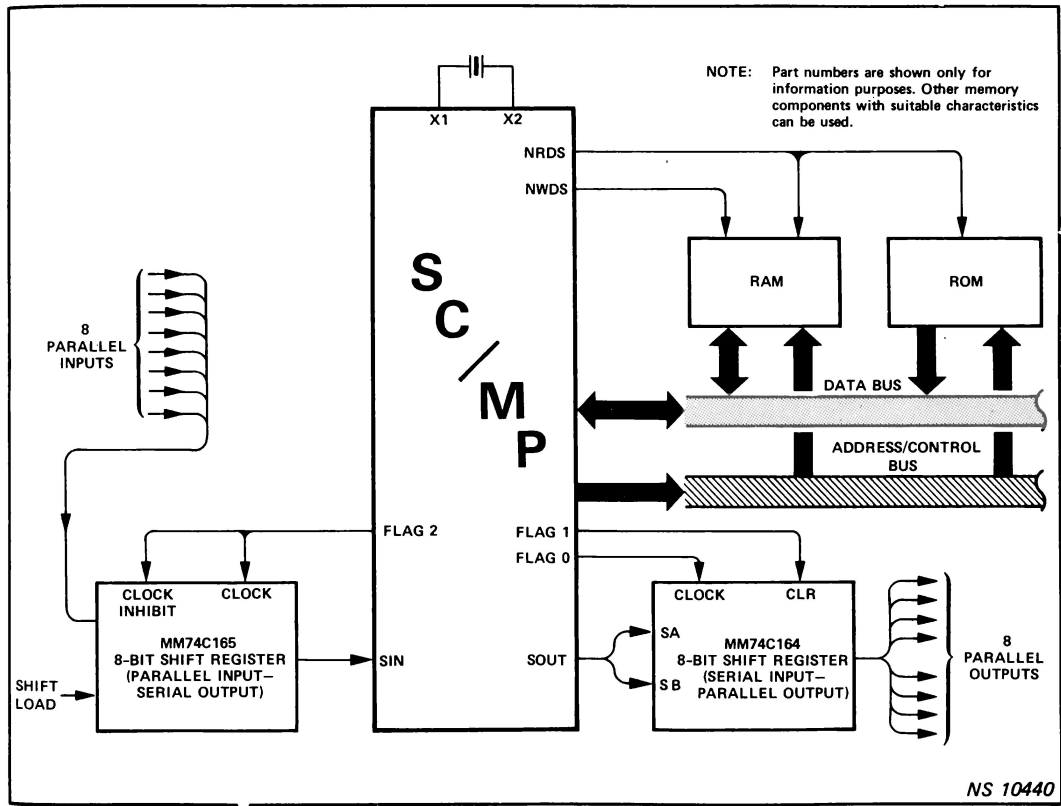



Fig. 2 - 11. Utilizzo di SC/MP con una semplice interfaccia seriale

2.3.6 Interrupt

Il sistema di interrupt di SC/MP è sotto controllo software ed opera come indicato in fig. 2-12. Prima di eseguire la fase di fetch, SC/MP testa il bit 3 dello status register. Se questo bit è a "0" (Interrupt enable basso) e la linea di continue è alta, il program counter viene incrementato, la nuova istruzione viene letta ed eseguita. Se il bit 3 è settato e la linea di Sense A è alta, viene servito l'interrupt: il bit 3 viene resettato ed il contenuto del program counter viene scambiato con il contenuto del registro puntatore 3. Evidentemente il puntatore 3 doveva contenere l'indirizzo della soubrutine che serve l'interruzione.

2.4 CONTROLLO E FLUSSO INTERNO DI DATI

2.4.1 Considerazioni generali

Tutti i trasferimenti di dati da e per SC/MP, avvengono tramite un bus di 8 bit bidirezionale di Input/Output (corrispondente ai pin da 9 a 16 di fig. 2-1). Le linee controllate dall'utente che permettono a SC/MP di operare sono due. Innanzitutto la CPU è inizializzata da uno stato "basso" ("0" logico) della linea NRST (reset) che duri almeno dodici periodi dell'oscillatore di SC/MP; poi NRST deve essere posta alta affinchè il processore possa operare. Cioè durante il normale funzionamento di SC/MP, la linea di NRST deve restare sempre alta.
La linea di CONT (continue) deve inoltre essere "alta" ("1" logico) affinchè il processore possa iniziare le proprie operazioni. A questo punto, il program counter viene incrementato e dalla locazione di memoria indicata da quest'ultimo (indirizzo x'0001) viene letto il primo byte (un'istruzione). Questo primo byte viene caricato nel registro delle istruzioni tramite il bus dei dati di Input/Output. L'istruzione viene quindi decodificata ed eseguita sotto controllo dalla logica di decodifica delle istruzioni e di controllo interno. Un'istruzione a singolo byte indica un'istruzione che può essere eseguita da SC/MP senza ulteriori riferimenti

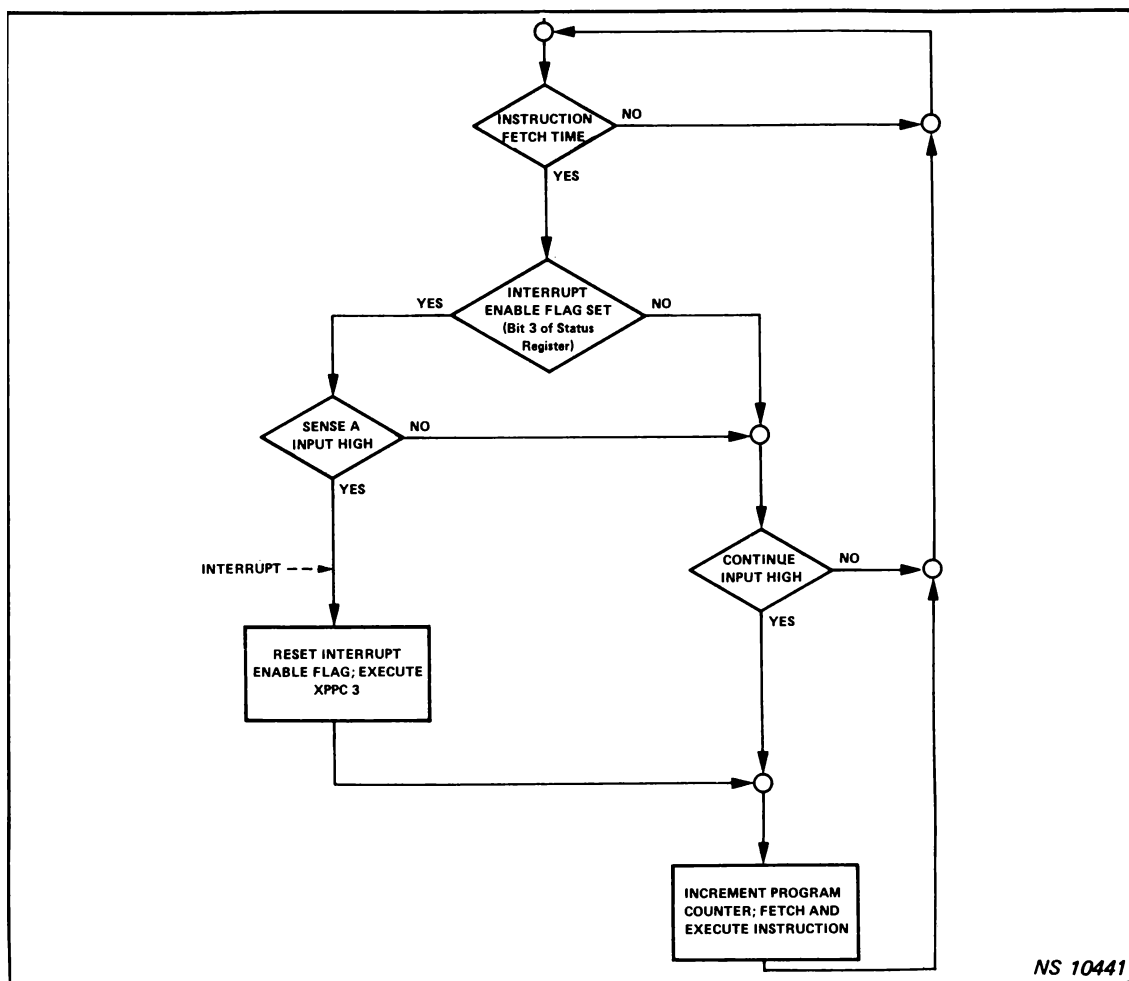



Fig. 2 -12. Flow della fase di fetch di SC/MP

in memoria. Questo tipo di Istruzioni hanno uno "0" come bit 7 (bit più significativo) mentre le istruzioni su due byte hanno un "1" come bit 7 del primo byte. Un'istruzione su due byte presenta, col primo byte, anche l'informazione se il secondo byte è un dato oppure un valore di spostamento (displacement). Quando il secondo byte è un dato, questo viene utilizzato da SC/MP durante l'esecuzione dell'istruzione. Quando invece il secondo byte è un campo di spostamento, esso viene utilizzato per calcolare un indirizzo, al quale SC/MP accederà durante l'esecuzione dell'istruzione per prelevare o per scrivere un dato. La manipolazione interna di dati, realizzata dalla sezione di controllo, dalla decodifica delle istruzioni e dall'unità aritmetico-logica, avviene in ogni caso mediante l'utilizzo di uno o più dei 7 registri disponibili al programmatore (vedi fig. 2-13). Tre di questi registri sono ad 8 bit; l'accumulatore, il registro degli stati ed il registro extension. Gli altri quattro registri sono a 16 bit e vengono chiamati puntatori: puntatore 0, 1, 2, 3. Il registro puntatore 0 è specializzato come program counter.

Tutti i registri a 16 bit sono connessi internamente a due bus di lettura e a due bus di scrittura, un bus di ciascun gruppo per la manipolazione del byte meno significativo (bit da 0 a 7) ed uno per il byte più significativo (bit da 8 a 15). I registri a 8 bit sono evidentemente connessi ad un solo bus di lettura e di scrittura. La possibilità di effettuare uno scambio tra il byte meno significativo e quello più significativo di un registro sarà descritta nell'appendice A con riferimento alle istruzioni appropriate. Oltre ai sette registri descritti esistono in SC/MP altri tre registri non accessibili al programmatore.

Due registri, il registro delle istruzioni e quello di Input/Output, sono a 8 bit. Il registro delle istruzioni memorizza il byte d'istruzione e lo presenta alla logica di decodifica e di controllo. Il registro di Input/Output dei dati è invece il legame tra il bus di Input/Output di SC/MP ed i suoi sette registri interni. Il registro degli indirizzi è invece a 16 bit ed è il legame tra i vari registri puntatori ed il bus di indirizzamento.

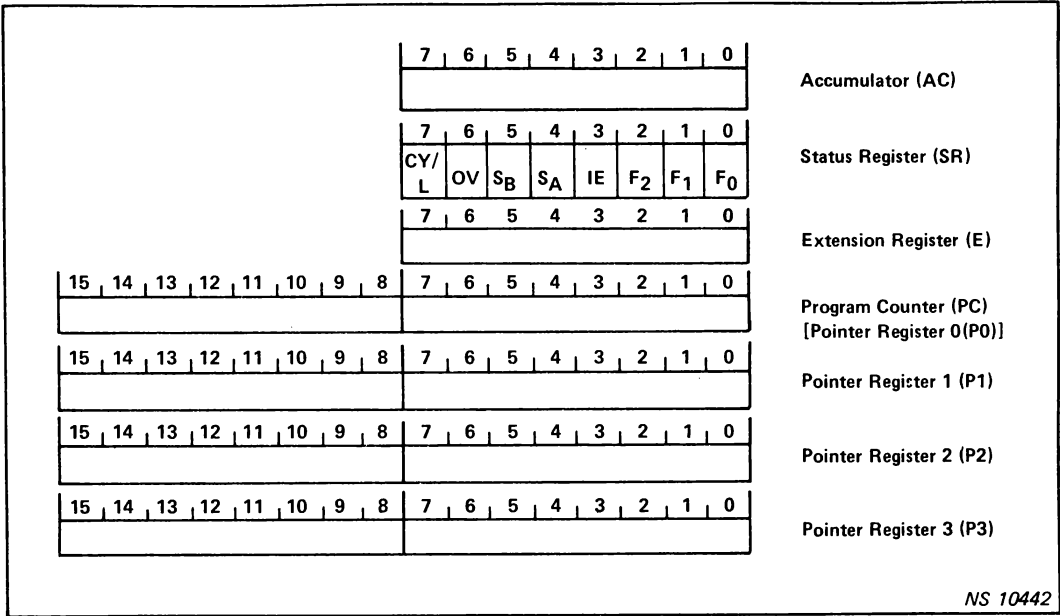



Fig. 2 - 13. Registri accessibili al programmatore

2.4.2 Descrizione dei registri di SC/MP

La fig. 2-13 mostra la struttura dei sette registri di SC/MP accessibili all'utente. Le funzioni di questi registri saranno invece descritte in dettaglio nei paragrafi 2.4.2.1 fino a 2.4.2.5.

2.4.2.1 Accumulatore (AC)

L'accumulatore (AC) è il principale registro di lavoro di SC/MP. Esso è utilizzato per realizzare le operazioni logiche od aritmetiche e come deposito del risultato delle stesse. Inoltre, i vari trasferimenti di dati e le operazioni di shift e di rotazione usano questo registro. Ben 37 istruzioni, delle 46 di SC/MP, utilizzano l'accumulatore.

2.4.2.2 Status Register (SR)

7	6	5	4	3	2	1	0	Posizione dei bit
CY/L	OV	SB	SA	IE	F2	F1	F0	Flags

Lo status register (SR) memorizza i flags di stato relativi sia ad operazioni aritmetiche che di controllo.

Bit *Descrizione*

0 Flag 0 (F0). E utilizzato dall'utente per funzioni di controllo sia software che hardware. Lo stato di questo flag è disponibile ad un corrispondente pin di SC/MP.

1 Flag 1 (F1). Come F0.

2 Flag 2 (F2). Come F0.

3 Interrupt enable (IE). SC/MP testa automaticamente la presenza di una richiesta di interruzione solo se questo flag è stato precedentemente settato.

4 Sense A (SA). Questo bit è collegato ad un pin di SC/MP e può essere utilizzato per testare delle condizioni esterne. Questo bit è a sola lettura, quindi non è possibile modificarne il valore via software. Quando il flag di interrupt enable è settato, sense A svolge la funzione di ingresso di richiesta di interrupt.

5 Sense B (SB). Come SA, tranne che non è utilizzabile come ingresso di richiesta di interruzione.

6 Overflow (OV). Questo bit viene settato quando si ha un overflow aritmetico durante le istruzioni di addizione (ADD, ADI o ADE) o di complemento ed addizione (CAD, CAI o CAE). Le istruzioni di somma decimate non lo modificano (DAD, DAI o DAE).

7 Carry/Link (CY/L). Questo bit viene settato se si ha un riporto dal bit più significativo durante l'esecuzione di un'istruzione di somma, complemento e somma o di somma decimale. Questo bit è inoltre utilizzato nelle istruzioni di rotazione con link o di shift con link (SRL o RRL).
CY/L è inoltre utilizzato come riporto in ingresso durante l'esecuzione delle istruzioni di add, complement-and-add e decimal-add.

2.4.2.3 Extension Register (E)

Il registro extension viene usato principalmente insieme all'accumulatore durante l'esecuzione delle istruzioni aritmetiche e logiche. Se il campo spostamento di un'istruzione con riferimento in memoria indicizzata è uguale a —128, il contenuto di E sostituisce il displacement stesso nel calcolo dell'indirizzo. Altra funzione del registro extension è quella di Input/Output seriale. Da notare che non è possibile scindere l'operazione di Input da quella di Output, cioè esse sono sempre realizzate contemporaneamente.

2.4.2.4 Program Counter (PC)

Il program counter coincide con il puntatore P0. Esso contiene l'indirizzo dell'istruzione che deve essere eseguita, e viene incrementato prima della fase di fetch. Il program counter opera come contatore solo nei suoi 12 bit meno significativi. Quindi, il contenuto del program counter ritorna a zero dopo aver indirizzato il byte 4.095.

2.4.2.5 Registri Puntatori (PTR)

Questi registri puntatori sono tre e sono disponibili per indirizzare sia memorie che periferiche, oppure come puntatori di stack, di pagina o registri indice.

2.4.3 Flusso dei dati tra i vari registri

La fig. 2-14 mostra come il contenuto dei vari registri interni di SC/MP possa essere manipolato dall'utente.

Note:

1 - Il byte meno signifivativo di ogni puntatore può essere scambiato con il contenuto dell'accumulatore.

2 - Il byte più significativo dei puntatori può essere scambiato con il contenuto dell'accumulatore.

3 - Il contenuto del puntatore 0 (program counter) può essere scambiato direttamente con quello di qualsiasi altro puntatore.

4 - L'accumulatore ed il registro extension possono scambiarsi i relativi contenuti; inoltre il contenuto dell'accumulatore può essere sostituito da quello del registro extension.

5 - I dati possono essere shiftati di un bit alla volta da e nel registro extension.

6 - Il contenuto dell'accumulatore può essere copiato nello status register e viceversa, il contenuto precedente del registro destinazione viene perso.

7 - Come mostrato, ogni istruzione con riferimento in memoria agisce anche sul contenuto dell'accumulatore.

2.4.4 Tipi di indirizzamento di SC/MP

2.4.4.1 Caratteristiche generali

Le istruzioni con riferimento in memoria possono utilizzare uno qualsiasi dei quattro registri puntatori visti precedentemente. Il program counter (puntatore 0) viene comunque utilizzato per indirizzare le istruzioni od i dati del programma. Inoltre, se non viene posta alcun'altra indicazione, il programma assemblato di SC/MP usa sempre il program counter per qualsiasi indirizzamento: quindi se non viene indicato alcun puntatore e l'accesso in memoria non può essere realizzato tramite P0, viene segnalato un errore di indirizzamento. La fig. 2-15a mostra come ciascun registro puntatore sia in grado di indirizzare in modo assoluto qualsiasi cella di memoria su 65.536, cioè da X'000 a X'FFFF.

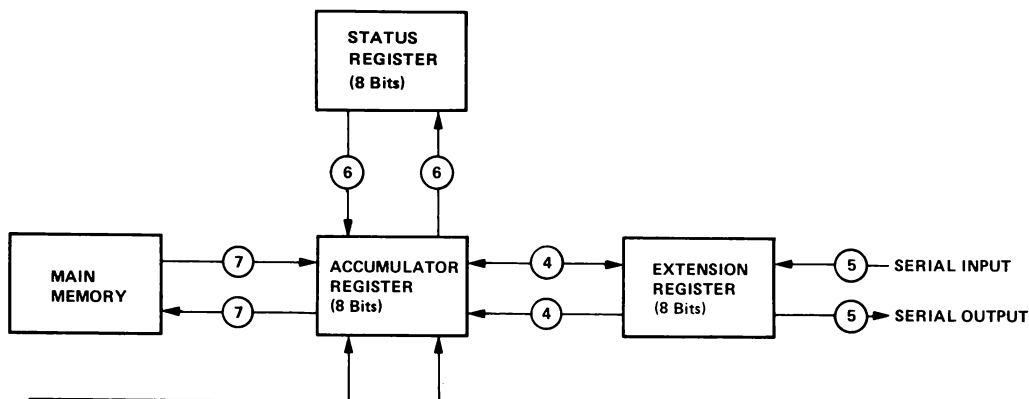



Fig. 2 - 14. Relazione tra i registri di SC/MP

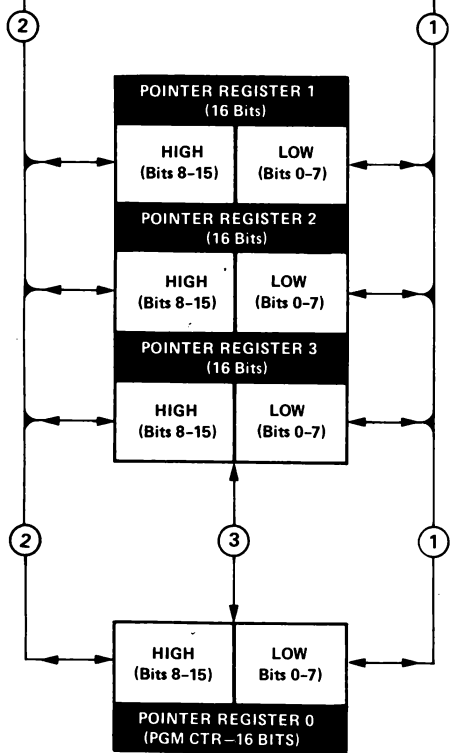
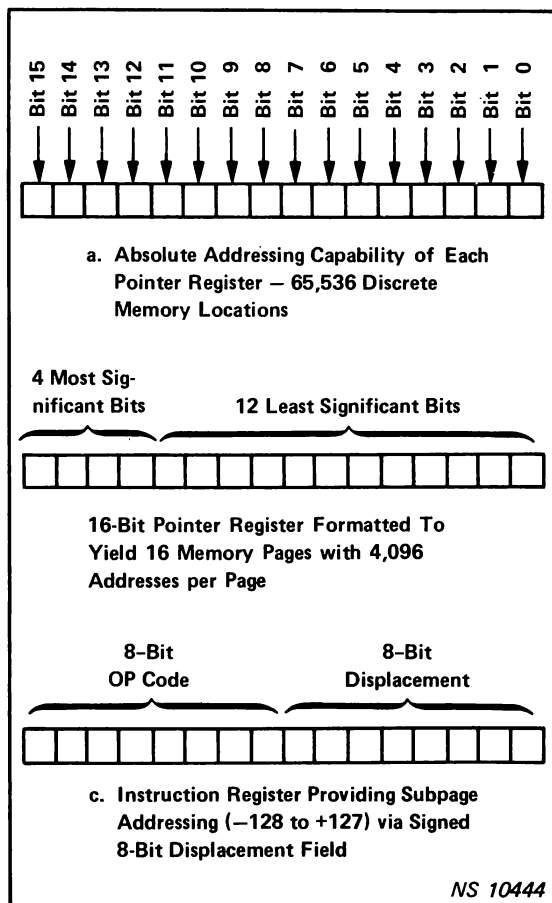



Fig. 2 - 15. Possibilità di indirizzamento di SC/MP

Per limitare ad un numero ragionevole di pin di SC/MP, la capacità d'indirizzamento assoluta dei registri è stata spezzata (vedi fig. 2-15b).
I quattro bit più significativi indicano la «pagina» e possono essere modificati solamente tramite un comando di load. I dodici bit meno significativi possono essere modificati tramite operazioni aritmetiche, indirizzando qualsiasi locazione delle 4.096 della pagina selezionata. Il byte di displacement di un'istruzione è invece un valore di 8 bit con segno che può essere sommato algebricamente con il contenuto di un qualsiasi puntatore, realizzando così una possibilità di spostamento da –128 a +127 locazioni.

2.4.4.2 Struttura della memoria

La memoria è organizzata come una sequenza di byte ed ogni byte è identificabile mediante un indirizzo di 16 bit il cui valore quindi può variare da 0 a X'FFFF (65.535). La fig. 2-16 mostra come questa memoria sia divisa in 16 pagine di 4.096 byte l'una.

Fig. 2 - 16. Struttura della memoria di SC/MP

Ogni indirizzo è quindi composto logicamente di due parti:
4 bit d'indirizzo di pagina e 12 bit di spostamento entro la pagina.

Quando si opera aritmeticamente per calcolare l'effettivo indirizzo di un dato, l'operazione viene eseguita solo sui 12 bit meno significativi e non si ha riporto sui 4 bit di indirizzamento della pagina. La seguente tabella mostra esempi delle varie situazioni che si possono creare:

	Lo spostamento è entro la pagina		Lo spostamento è eccessivo	
	Indirizzo nella pagina	Pagina	Indirizzo nella pagina	Pagina
Indirizzo attuale	0	FB4	0	FB4
Displacement dell'istruzione	–	05	–	4d
Nuovo Indirizzo	0	FB9	0	001

L'incremento del program counter per la lettura di una nuova istruzione segue le medesime regole sopra viste, quindi anche in questo caso non si ha riporto sui 4 bit più significativi.
Conseguentemente se un'istruzione a 2 byte è posta in memoria in modo tale da essere scritta su due pagine, si ha un errore.

Ad esempio si supponga che in memoria, all'indirizzo X1FFF, sia scritto X' C0A2 (LD 20A2). Tuttavia il processore leggerà (avendo PC=X'1FFF) X' C081, non eseguendo automaticamente il passaggio di pagina e quindi leggendo le celle X' 1FFF e X'1000. Sarà quindi l'utilizzatore che dovrà al momento della stesura del programma, prendere le precauzioni necessarie affinchè ciò non avvenga.

2.4.4.3 Format delle istruzioni con riferimento in memoria

Il format delle istruzioni che fanno riferimento alla memoria (o alle periferiche) è il seguente:

Le istruzioni con riferimento in memoria usano i metodi di indirizzamento relativo al PC, indicizzato o autoindicizzato.
Le istruzioni di incremento/decremento di una cella di memoria e quelle di trasferimento utilizzano il metodo di indirizzamento relativo al PC o quello indicizzato. L'indirizzamento immediato è invece relativo alle specifiche istruzioni (ad es. LDI, ANI, ...).
I vari metodi di indirizzamento sono mostrati nella seguente tabella:

Tipo di indirizzamento	Formato dell'istruzione		
	m	ptr.	disp.
relativo al P. C.	0	0	da −128 a +127
indicizzato	0	1,2 o 3	da −128 a +127
immediato	1	0	da −128 a +127
auto-indicizzato	1	1,2 o 3	da −128 a +127

È importante ricordare, comunque, che nel caso di riferimento in memoria relativo al PC, autoindicizzato, se il displacement ha valore -128, il contenuto del registro extension sostituisce il displacement stesso nel calcolo dell'indirizzo effettivo.

2.4.4.3.1 Indirizzamento relativo al P.C.

L'indirizzamento relativo al program counter è realizzato sommando lo spostamento (displacement), indicato dal campo operando di un'istruzione, al contenuto corrente del program counter.
Il displacement è un numero a 8 bit, con segno e logica complemento a due; conseguentemente si può indirizzare una zona di memoria che va da −128 a +127 celle rispetto al valore di P.C.
Da notare che il program counter, durante l'esecuzione di un'istruzione, contiene il valore dell'ultimo byte dell'istruzione in corso e che verrà comunque incrementato prima di eseguire l'istruzione successiva.

2.4.4.3.2 Indirizzamento immediato

Nell'indirizzamento immediato il secondo byte dell'istruzione è l'operando da utilizzare nell'esecuzione dell'istruzione stessa. Si può quindi notare la differenza tra l'istruzione di Load (LD) da quella di Load Immediato (LDI). L'istruzione di Load usa il contenuto del secondo byte dell'istruzione per il calcolo dell'indirizzo della cella di memoria da cui caricare il dato L'istruzione di Load Immediato utilizza invece il contenuto del secondo byte dell'istruzione stessa come dato da caricare in accumulatore.

2.4.4.3.3 Indirizzamento indicizzato

Nel caso di indirizzamento, il displacement viene sommato al contenuto del registro puntatore indicato nell'istruzione stessa ed il risultato è l'indirizzo effettivo. Il contenuto del registro puntatore resta inalterato. È evidente che in questo modo l'utente può accedere a qualsiasi cella di memoria, anche al di fuori del range accessibile mediante un indirizzamento relativo al P.C.

2.4.4.3.4 Indirizzamento indicizzato

L'indirizzamento autoindicizzato è sostanzialmente identico a quello indicizzato, ma fornisce al programmatore la possibilità di modificare il registro puntatore indicato dal valore del displacement. Se il displacement è minore di zero, il registro puntatore viene decrementato prima dell'operazione di scrittura o lettura del dato. Se il displacement è maggiore od uguale a zero, il registro puntatore viene incrementato dopo l'operazione di accesso alla cella contenente il dato.

Appendice A

SET DI ISTRUZIONI DI SC/MP

A. 1 Introduzione.

Le tabelle A-1, A-2 e A-3 definiscono le istruzioni di SC/MP nei termini di simboli e notazioni, format delle istruzioni con riferimento in memoria ed un sommario descrittivo delle stesse.
La fig. A-1 invece mostra l'occupazione del bus relativamente all'esecuzione delle varie istruzioni di SC/MP. Queste informazioni sono particolarmente utili nel caso di applicazione di più processori o di logiche di DMA nello stesso sistema.

Tabella A -1

Simbolo o notazione	Significato
AC	Accumulatore.
CY/L	Flag di Carry o Link.
data	Dato di 8 bit, corrispondente al secondo byte di un'istruzione di tipo immediato.
disp	Displacement; rappresenta un operando (data) nelle istruzioni senza riferimento in memoria, oppure un valore utile al calcolo di un indirizzo nel caso di riferimento in memoria. E in ogni caso un numero con segno, complemento a due.
EA	Indica l'indirizzo effettivo specificato dall'istruzione.
E	Registro Extension; è un registro di lavoro.
i	Bit non precisato di un registro.
IE	Flag di abilitazione dell'interrupt.
m	Bit di modo, presente nel codice operando delle istruzioni con riferimento in memoria. Se è ad 1, indica @.
OV	Flag di overflow.
PC	Program counter (registro puntatore 0); durante il calcolo dell'indirizzo effettivo EA, esso punta all'ultimo byte dell'istruzione in esecuzione.
ptr	Indica un registro puntatore (ptr = da 0 a 3). Viene indicato dal primo byte dell'istruzione.
$ptr_{n:m}$	Gruppo di bit di un registro puntatore; n : m = 7 : 0 o 15 : 8.
SIN	Indica il pin di serial input.
SOUT	Indica il pin di serial out.
SR	Indica il registro di stato.
( )	Significa il «contenuto di». Ad esempio (EA) è il contenuto della cella indicata dall'indirizzo effettivo EA.
[ ]	Indica un campo opzionale nella notazione assembler di un'istruzione.
~	Indica il complemento ad 1 del valore a destra del simbolo.
→	Significa «sostituisce».
←	Significa «è sostituito da».
←→	Significa «scambio tra».
@	Quando viene usato nel campo operando di un'istruzione, setta il bit di modo m ad 1, ad indicare un indirizzamento auto-indicizzato (con incremento:decremento del Puntatore).
10^+	Somma in modulo 10
$\wedge$	And
V	OR
$\forall$	Exclusive-OR.
$\geqq$	Maggiore od uguale a.
=	Eguale.
$\neq$	Non eguale.

Tabella A-2. Format delle istruzioni con riferimento in memoria				
Indirizzamento	Formato dell'operando			
	Traduzione ottenuta dall'Assembler			Notazione in linguaggio Assembler
	m	ptr	disp*	
Relativo al PC	0	0	da −128 a +127	disp
Indicizzato	0	1,2 o 3	da -128+127	disp (ptr)
Auto-indicizzato	1	1,2 o 3	da −128 a +127	disp (ptr)

* Se disp = −128, (E) sostituisce (disp).

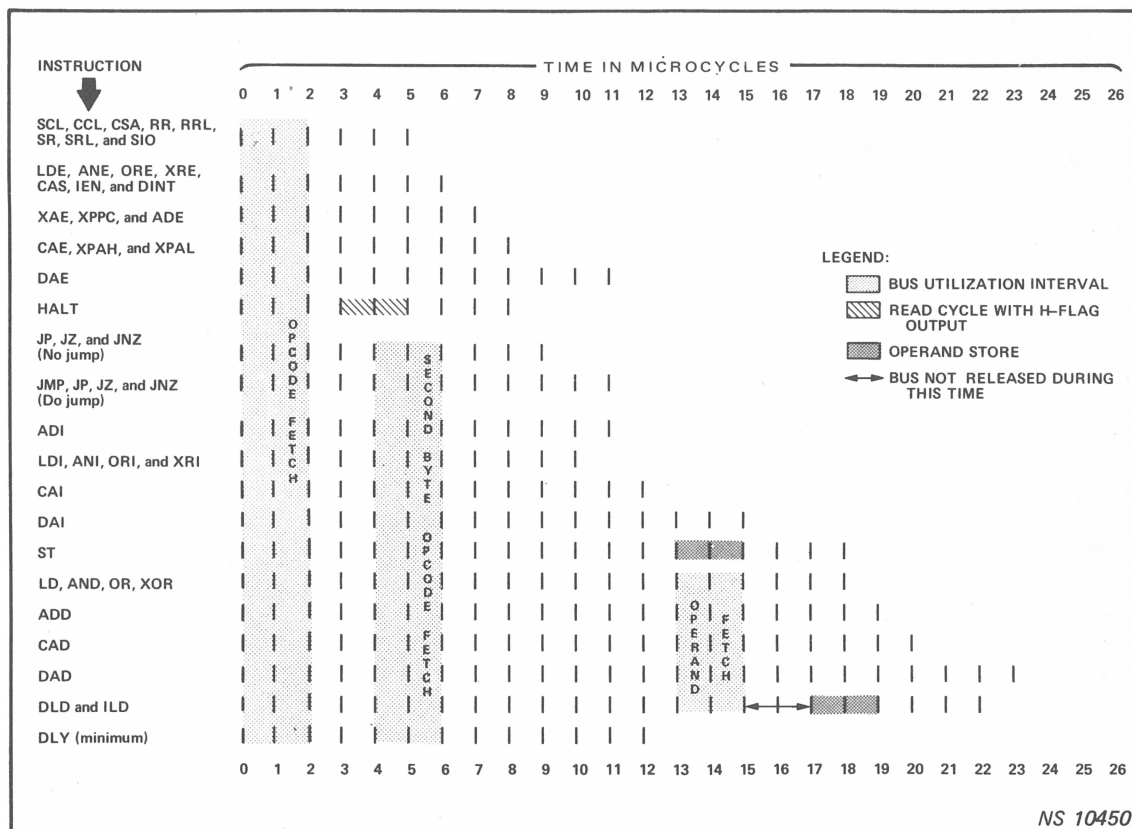



Fig. A-1. Utilizzazione del bus da parte di ciascuna istruzione.

Tabella A-3 - Istruzioni di SC/MP			
Istruzione	Operazione	Format	Tempo di esecuzione (microcicli)
Istruzioni con riferimento in memoria			
LOAD LD (C0) BYTE 1: bit 7…3 = 1 1 0 0 0; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(AC) \leftarrow (EA)$ Il contenuto dell'accumulatore (AC) viene sostituito dal contenuto dell'indirizzo effettivo (EA). Il contenuto precedente di AC è perso, il contenuto di EA resta inalterato.	LD disp (ptr) disp @ (ptr) disp	18
STORE ST (C8) BYTE 1: bit 7…3 = 1 1 0 0 1; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(EA) \leftarrow (AC)$ Il contenuto dell'indirizzo effettivo (EA) viene sostituito dal contenuto dell'accumulatore (AC). Il contenuto iniziale di EA è perso, il contenuto di AC resta inalterato.	ST disp (ptr) disp @ (ptr) disp	18
AND AND (D0) BYTE 1: bit 7…3 = 1 1 0 1 0; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(AC) \leftarrow (AC) \wedge (EA)$ Viene realizzato l'AND tra il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, il contenuto di EA non è modificato.	AND disp (ptr) disp @ (ptr) disp	18
OR OR (D8) BYTE 1: bit 7…3 = 1 1 0 1 1; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(AC) \leftarrow (AC) \vee (EA)$ Viene realizzato l'OR tra il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, quello di EA non è modificato.	OR disp (ptr) disp @ (ptr) disp	18
EXCLUSIVE-OR XOR (E0) BYTE 1: bit 7…3 = 1 1 1 0 0; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(AC) \leftarrow (AC) \veebar (EA)$ Viene realizzato l'EXCLUSIVE-OR tra il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, quello di EA non è modificato	XOR disp (ptr) disp @ (ptr) disp	18
DECIMAL ADD DAD (E8) BYTE 1: bit 7…3 = 1 1 1 0 1; bit 2 = m; bit 1 0 = ptr BYTE 2: bit 7…0 = disp	$(AC) \leftarrow (AC)_{10} + (EA)_{10} + (CY/L)$; CY/L Il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) sono considerati come due numeri maggiori od eguali a 0 e inferiori o eguali a 99 espressi su 2 digit BCD (Binary-Coded-Decimal). Il contenuto di AC e di EA	DAD disp (ptr) disp @ (ptr) disp	23

segue

Tabella A-3

	ed il carry (CY/L) sono sommati ed il risultato è posto in AC. Il contenuto iniziale di AC è perso, il contenuto di EA resta inalterato. Il flag di overflow non viene interessato, mentre quello di carry è settato in funzione del riporto ottenuto dal digit più significativo.		
ADD — ADD (F0) BYTE 1 (bit 7 … 3, 2, 1 0) — BYTE 2 (bit 7 … 0) 1 1 1 1 0 | m | ptr | disp	(AC) ← (AC) + (EA) + (CY/L); CY/L, OV Il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) sono considerati come due numeri binari di 8 bit, logica complemento a due. Il contenuto dell'accumulatore (AC), dell'indirizzo effettivo (EA) ed il carry (CY/L) sono sommati algebricamente ed il risultato è posto in AC. Il flag di carry nello status register viene settato a 1 se si ha riporto dal bit più significativo, altrimenti viene resettato. Il flag di overflow (OV) nello status register viene settato quando si ha overflow, cioè quando il segno del risultato è diverso da quello di entrambi gli operandi.	ADD disp (ptr) disp @ (ptr) disp	19
COMPLEMENT AND ADD — CAD (F8) BYTE 1 (bit 7 … 3, 2, 1 0) — BYTE 2 (bit 7 … 0) 1 1 1 1 1 | m | ptr | disp	(AC) ← (AC) + ~ (EA) + (CY/L). CY/L, OV Il contenuto dell'accumulatore (AC) ed il contenuto dell'indirizzo effettivo (EA) sono considerati come due numeri binari di 8 bit. Il contenuto dell'accumulatore (AC), il complemento ad uno del contenuto dell'indirizzo effettivo (EA) ed il carry (CY/L) sono sommati algebricamente ed il risultato è posto in AC. Il contenuto iniziale di AC viene perso, il contenuto di EA resta inalterato. Il flag di carry (CY/L) nello status register viene settato se si ha riporto dal bit più significativo, altrimenti viene resettato. Il flag di overflow (OV) nello status register viene settato se il segno del risultato è uguale al segno di (EA) ed opposto al segno di (AC), altrimenti è resettato	CAD disp (ptr) disp @(ptr) disp	20
Istruzioni di tipo immediato			
LOAD IMMEDIATE — LDI (C4) BYTE 1 (bit 7 … 0) — BYTE 2 (bit 7 … 0) 1 1 0 0 0 1 0 0 | data	(AC) ← (data) Il contenuto dell'accumulatore (AC) viene sostituito dal byte di dato (secondo byte dell'istruzione) Il contenuto iniziale di AC viene perso, il byte di dato non viene modificato.	LDI data	10

segue

Tabella A-3

Istruzione	Descrizione		
AND IMMEDIATE ANI (D4) BYTE 1: 7…0 = 1 1 0 1 0 1 0 0 BYTE 2: 7…0 = data	(AC) ← (AC) ∧ (data) Il contenuto dell'accumulatore (AC) viene posto in AND col byte di dato ed il risultato viene posto in AC Il contenuto iniziale di AC viene perso, il byte di dato resta inal terato.	ANI data	10
OR IMMEDIATE ORI (DC) BYTE 1: 7…0 = 1 1 0 1 1 1 0 0 BYTE 2: 7…0 = data	(AC) ← (AC) ∨ (data) Il contenuto dell'accumulatore (AC) viene posto in OR col byte di dato ed il risultato è posto in AC. Il contenuto iniziale di AC viene perso, il byte di dato resta inalterato	ORI data	10
EXCLUSIVE-OR IMMEDIATE XRI (E4) BYTE 1: 7…0 = 1 1 1 0 0 1 0 0 BYTE 2: 7…0 = data	(AC) ← (AC) ∀ (data) Il contenuto dell'accumulatore (AC) viene posto in EXCLUSIVE-OR col byte di dato ed il risultato è posto in AC. Il contenuto iniziale di AC viene perso, il byte di dato resta inalterato	XRI data	10
DECIMAL ADD IMMEDIATE DAI (EC) BYTE 1: 7…0 = 1 1 1 0 1 1 0 0 BYTE 2: 7…0 = data	(AC) ← $(AC)_{10}$ + $(data)_{10}$ + (CY/L); CY/L Il contenuto dell'accumulatore (AC) ed il byte di dato sono considerati come 2 numeri maggiori o eguali a 0 e inferiori o eguali a 99 espressi su 2 digit BCD (Binary-Coded-Decimal). Il contenuto di AC, il byte di dato ed il carry sono sommati ed il risultato è posto in AC. Il contenuto iniziale di AC è perso, il byte di dato resta inalterato. Il flag di overflow non viene interessato, mentre quello di carry è settato in funzione del riporto ottenuto dal digit più significativo.	DAI data	15
ADD IMMEDIATE ADI (F4) BYTE 1: 7…0 = 1 1 1 1 0 1 0 0 BYTE 2: 7…0 = data	(AC) ← (AC) + (data) + (CY/L); CY/L, OV Il contenuto dell'accumulatore (AC) ed il byte di dato sono considerati come 2 numeri binari di 8 bit, logica complemento a due. Il contenuto dell'accumulatore (AC), il byte di dato ed il carry sono sommati algebricamente ed il risultato è posto in AC. Il flag di carry nello status register viene settato ad 1 se si ha riporto dal bit più significativo, altrimenti viene resettato. Il flag di overflow (OV) nello status register viene settato quando si ha overflow, cioè quando il segno del risultato è diverso da quello di entrambi gli operandi.	ADI data	11

segue

Tabella A-3

Istruzione	Operazione	Formato	Cicli
COMPLEMENT AND ADD IMMEDIATE — CAI (FC) BYTE 1: 7…0 — 1 1 1 1 1 1 0 0 BYTE 2: 7…0 — data	(AC) ← (AC) + ~ (data) + (CY/L); CY/L, OV Il contenuto dell'accumulatore (AC) ed il byte di dato sono considerati come 2 numeri binari di 8 bit. Il contenuto dell'accumulatore (AC) il complemento ad uno del byte di dato ed il carry sono sommati algebricamente ed il risultato è posto in AC. Il contenuto iniziale di AC viene perso, il byte di dato resta inalterato. Il flag di carry nello status register viene settato se si ha riporto dal bit più significativo, altrimenti viene resettato.	CAI data	12
Istruzioni di trasferimento			
JUMP — JMP (90) BYTE 1: 7…2 — 1 0 0 1 0 0; 1 0 — ptr BYTE 2: 7…0 — disp	(PC) ← EA L'indirizzo effettivo (EA) sostituisce il contenuto del program counter (PC). La prossima istruzione viene letta dalla locazione PC + 1.	JMP disp (ptr)	11
JUMP IF POSITIVE — JP (94) BYTE 1: 7…2 — 1 0 0 1 0 1; 10 — ptr BYTE 2: 7…0 — disp	IF (AC) ≧ 0, (PC) ← EA Se il contenuto dell'accumulatore (AC) è positivo o zero, l'indirizzo effettivo (EA) sostituisce il contenuto del program counter (PC). La prossima istruzione viene letta dalla locazione PC + 1.	JP disp (ptr)	9 (no jump); 11 (jump)
JUMP IF ZERO — JZ (98) BYTE 1: 7…2 — 1 0 0 1 1 0; 1 0 — ptr BYTE 2: 7…0 — disp	IF (AC) = 0, (PC) ← EA Se il contenuto dell'accumulatore (AC) è zero, l'indirizzo effettivo (EA) sostituisce il contenuto del program counter (PC). La prossima istruzione viene letta dalla locazione PC + 1.	JZ disp (ptr)	9 (no jump); 11 (jump)
JUMP IF NOT ZERO — JNZ (9C) BYTE 1: 7…2 — 1 0 0 1 1 1; 1 0 — ptr BYTE 2: 7…0 — disp	IF (AC) ≠ 0, (PC) ← EA Se il contenuto dell'accumulatore (AC) non è zero, l'indirizzo effettivo (EA) sostituisce il contenuto del program counter (PC). La prossima istruzione viene letta dalla locazione PC + 1.	JNZ disp (ptr)	9 (no jump); 11 (jump)
Istruzioni di incremento/decremento in memoria			
INCREMENT AND LOAD — ILD (A8) BYTE 1: 7…2 — 1 0 1 0 1 0; 1 0 — ptr BYTE 2: 7…0 — disp	(AC), (EA) ← (EA) + 1 Il contenuto dell'indirizzo effettivo (EA) viene incrementato di 1 ed il risultato viene posto nell'accumulatore (AC) ed anche in EA. Il contenuto iniziale di AC e di EA è perso. I flags di carry e di overflow non sono modificati.	ILD disp disp (ptr)	22

segue

Tabella A-3

DECREMENT AND LOAD DLD (B8) BYTE 1: bit 7 … 2 | 1 0 — BYTE 2: bit 7 … 0 1 0 1 1 1 0 | ptr | disp	$(AC), (EA) \leftarrow (EA) - 1$ Il contenuto dell'indirizzo effettivo viene decrementato di 1 ed il risultato viene posto nell'accumulatore (AC) ed anche in EA. Il contenuto iniziale di AC e di EA è perso. I flags di carry e di overflow non sono modificati.	DLD disp disp (ptr)	22
Istruzioni relative al registro Extension			
LOAD FROM EXTENSION LDE (40) 7 … 0 0 1 0 0 0 0 0 0	$(AC) \leftarrow (E)$ Il contenuto dell'accumulatore (AC) viene sostituito dal contenuto del registro extension (E). Il contenuto iniziale di AC viene perso; il contenuto di E resta inalterato.	LDE	6
EXCHANGE AC AND EXTENSION XAE (01) 7 … 0 0 0 0 0 0 0 0 1	$(AC) \longleftrightarrow (E)$ Il contenuto dell'accumulatore (AC) viene scambiato col contenuto del registro extension (E).	XAE	7
AND EXTENSION ANE (50) 7 … 0 0 1 0 1 0 0 0 0	$(AC) \leftarrow (AC) \wedge (E)$ Il contenuto dell'accumulatore (AC) viene posto in AND con il contenuto del registro extension (E) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, il contenuto di E resta inalterato.	ANE	6
OR EXTENSION ORE (58) 7 … 0 0 1 0 1 1 0 0 0	$(AC) \leftarrow (AC) \vee (E)$ Il contenuto dell'accumulatore (AC) vien posto in OR con il contenuto del registro extension (E) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, il contenuto di E resta inalterato.	ORE	6
EXCLUSIVE - OR EXTENSION XRE (60) 7 … 0 0 1 1 0 0 0 0 0	$(AC) \leftarrow (AC) \veebar (E)$ Il contenuto dell'accumulatore (AC) viene posto in EXCLUSIVE-OR con il contenuto del registro extension (E) ed il risultato viene posto in AC. Il contenuto iniziale di AC è perso, il contenuto di E resta inalterato.	XRE	6
DECIMAL ADD EXTENSION DAE (68) 7 … 0 0 1 1 0 1 0 0 0	$(AC) \leftarrow (AC)_{10} + (E)_{10} + (CY/L)$; CY/L Il contenuto dell'accumulatore (AC) ed il contenuto del registro extension (E) sono considerati come due numeri maggiori od eguali a 0 ed inferiori od eguali a 99 espressi su	DAE	11

segue

Tabella A-3

	due digit (BCD) (Binary-Coded-Decimal). Il contenuto di AC, di E ed il carry (CY/L) sono sommati e il risultato è posto in AC. Il contenuto iniziale di AC è perso, il contenuto di E resta inalterato. Il flag di overflow non viene interessato mentre quello di carry è settato in funzione del riporto ottenuto dal digit più significativo.		
ADD EXTENSION ADE (70) 7 0 0 1 1 1 0 0 0 0	(AC) ← (AC) + (E) + (CY/L); CY/L, (CY/L), OV Il contenuto dell'accumulatore (AC) ed il contenuto del registro extension (E) sono considerati come due numeri binari di 8 bit, logica complemento a due. Il contenuto dello accumulatore (AC), del registro extension (E) ed il carry (CY/L) sono sommati algebricamente ed il risultato è posto in AC. Il flag di carry nello status register viene settato a 1 se si ha riporto dal bit più significativo altrimenti viene resettato. Il flag di overflow (OV) nello status register viene settato quando si ha overflow, cioè quando il segno del risultato è diverso da quello di entranbi gli operandi.	ADE	7 8
COMPLEMENT AND ADD EXTENSION CAE (78) 7 0 0 1 1 1 1 0 0 0	(AC) ← (AC) + ~ (E) + (CY/L); CY/L, OV Il contenuto dell'accumulatore (AC) ed il contenuto del registro extension (E) sono considerati come due numeri binari di 8 bit. Il contenuto dell'accumulatore (AC), il complemento ad uno del contenuto del registro extension (E) ed il carry (CY/L) sono sommati algebricamente ed il risultato è posto in AC. Il contenuto iniziale di AC viene perso, il contenuto di E resta inalterato. Il flag di carry (CY/L) nello status register viene settato se si ha riporto dal bit più significativo, altrimenti viene resettato. Il flag di overflow (OV) nello status register viene settato se il segno del risultato è uguale al segno di (AC), altrimenti è resettato.	CAE	8
Istruzioni di scambio sui pointers			
EXCHANGE POINTER LOW XPAL (30) 7 2 | 1 0 0 0 1 1 0 0 | ptr	(AC) ←→ ($PTR_{7 \div 0}$) Il contenuto dell'accumulatore (AC) viene scambiato con il byte meno significativo (bit 7÷0) del registro puntatore indicato (PTR)	XPAL ptr	8

segue

Tabella A-3

EXCHANGE POINTER HIGH — XPAH (34) 7 2 | 1 0 0 0 1 1 0 1 | ptr	(AC) ⟷ ($PTR_{15:8}$) Il contenuto dell'accumulatore (AC) viene scambiato con il byte più significativo (bit 15÷8) del registro puntatore indicato (PTR)	XPAH ptr	8
EXCHANGE POINTER WITH PC — XPPC (3C) 7 2 | 1 0 0 0 1 1 1 1 | ptr	(PC) ⟷ (PTR) Il contenuto del Program counter (PC) viene scambiato con il contenuto del registro puntatore indicato (PTR).	XPPC ptr	7
Istruzioni di shift, rotazione, serial Input/Output			
SERIAL INPUT/OUTPUT — SIO (19) 7 0 0 0 0 1 1 0 0 1	(E_i) → (E_{i-1}), SIN → (E_7), (E_0) → SOUT Il contenuto del registro extension (E) viene shiftato a destra di una posizione. Il contenuto iniziale del bit 0 viene presentato sul pin di output SOUT. Il dato presente al pin di ingresso SIN viene caricato nel bit 7 di E.	SIO	5
SHIFT RIGHT — SR (1C) 7 0 0 0 0 1 1 1 0 0	(AC_i) → (AC_{i-1}), 0 → (AC_7) Il contenuto dell'accumulatore (AC) viene shiftato a destra di una posizione. Il contenuto iniziale del bit 0 viene perso. Nel bit 7 entra uno 0.	SR	5
SHIFT RIGHT WITH LINK — SRL (1D) 7 0 0 0 0 1 1 1 0 1	(AC_i) → (AC_{i-1}), (CY/L) → (AC_7) Il contenuto dell'accumulatore viene shiftato a destra di una posizione. Il contenuto iniziale del bit 0 viene perso. Il flag di link (CY/L) dello status register viene caricato nel bit 7. Il flag di link non viene alterato.	SRL	5
ROTATE RIGHT — RR (1E) 7 0 0 0 0 1 1 1 1 0	(AC_i) → (AC_{i-1}), (AC_0) → (AC_7) Il contenuto dell'accumulatore (AC) viene ruotato verso destra di una posizione. Il contenuto iniziale del bit 0 viene posto nel bit 7.	RR	5
ROTATE RIGHT WITH LINK — RRL (1F) 7 0 0 0 0 1 1 1 1 1	(AC_i) → (AC_{i-1}), (AC_0) → (CY/L) → (AC_7) Il contenuto dell'accumulatore (AC) viene ruotato verso destra di una posizione. Il contenuto iniziale del bit 0 viene posto nel flag di link (CY/L) dello status register, mentre il contenuto iniziale del flag di link viene caricato nel bit 7 di AC.	RRL	5

segue

Tabella A-3

Istruzioni varie				
HALT 7 0 0 0 0 0 0 0 0 0	HALT (00)	Il flag di Halt viene posto ad 1 durante la fase di Output della parola di stato. Questa istruzione si presta, per particolari applicazioni, a funzioni diverse da quelle intrinseche all'istruzione di Halt.	HALT	8
CLEAR CARRY LINK 7 0 0 0 0 0 0 0 1 0	CCL (02)	(CY/L) ← 0 Il flag di carry/link presente nello status register è azzerato. Gli altri bit dello status register (SR) non sono modificati.	CCL	5
SET CARRY LINK 7 0 0 0 0 0 0 0 1 1	SCL (03)	(CY/L) ← 1 Il flag di carry/link presente nello status register viene settato. Gli altri bit dello status register (SR) non sono modificati.	SCL	5
ENABLE INTERRUPT 7 0 0 0 0 0 0 1 0 1	IEN (05)	(IE) ← 1 Il flag interrupt enable (IE) presente nello status register viene settato. Gli altri bit dello status register (SR) non sono modificati. Viene abilitato il sistema di interrupt del processore. Il processore accetterà un eventuale segnale di interruzione dopo la fase di fetch e di esecuzione della istruzione successiva alla IEN.	IEN	6
DISABLE INTERRUPTS 7 0 0 0 0 0 0 1 0 0	DINT (04)	(IEN) ← 0 Il flag interrupt enable (IE) presente nello status register è azzerato. Gli altri bit dello status register (SR) non sono modificati. Il sistema di interrupt del processore è disabilitato. Eventuali richieste di interruzione non verranno prese in considerazione dal microprocessore.	DINT	6
COPY STATUS TO AC 7 0 0 0 0 0 0 1 1 0	CSA (06)	(AC) ← (SR) Il contenuto dell'accumulatore (AC) è sostituito dal contenuto dello status register (SR). Il contenuto iniziale di AC è perso, il contenuto di SR resta inalterato.	CSA	5

segue

Tabella A-3

COPY AC TO STATUS 7 … 0 0 0 0 0 0 1 1 1	CAS (07)	(SR) ← (AC) Il contenuto dell'accumulatore (AC) sostituisce il contenuto dello status register (SR). I bit 4 e 5 di SR, cioè sense A e sense B, non sono modificati da questa istruzioni. Il contenuto iniziale di SR, a parte i due bit di sense, è perso. Il contenuto dell'accumulatore non viene modificato. Se IE viene posto ad 1 da questa istruzione, il sistema di interruzione sarà abilitato dopo le fasi di fetch e di esecuzione dell'istruzione successiva alla CAS.	CAS	6
NO OPERATION 7 … 0 0 0 0 0 1 0 0 0	NOP (08)	(PC) ← (PC) + 1 Il program counter (PC) è incrementato di 1. L'istruzione NOP necessita di un tempo di esecuzione minimo di 5 microcicli. Eventuali codici operativi non definiti incontrati, sono considerati come istruzioni di NOP su uno o due byte, a seconda della lunghezza dell'istruzione non definita. Si ottengono in questo modo NOP da 5 a 10 microcicli, in dipendenza del codice errato.	NOP	5 (min) 10 (max)
DELAY BYTE 1: 7 … 0 / 1 0 0 0 1 1 1 1 BYTE 2: 7 … 0 / disp	DLY (8F)	DELAY = 13 + 2 (AC) + 2 disp + 2^9 disp Questa istruzione permette un delay del microprocessore di lunghezza variabile. Il contenuto dell'accumulatore (AC) e del campo spostamento (disp) sono considerati numeri binari senza segno quindi con valore massimo 255. Il tempo di esecuzione in termini di microcicli è determinato dal numero dato da una precisa equazione. La tabella seguente fornisce alcuni tempi tipici di esecuzione. Il range del delay va da 13 a 131593 microcicli.	DLY	13 (min) 131593 (max)

	AC									
	0	25	50	75	100	125	150	175	200	225
0	13	63	113	163	213	263	313	363	413	463
1	527	577	627	677	727	777	827	877	927	977
2	1041	1091	1141	1191	1241	1291	1341	1391	1441	1491
3	1555	1605	1655	1705	1755	1805	1855	1905	1955	2005
4	2069	2119	2169	2219	2269	2319	2369	2419	2469	2519
5	2583	2633	2683	2733	2783	2833	2883	2933	2983	3033
6	3097	3147	3197	3247	3297	3347	3397	3447	3497	3547
7	3611	3661	3711	3761	3811	3861	3911	3961	4011	4061
8	4125	4175	4225	4275	4325	4375	4425	4475	4525	4575
9	4639	4689	4739	4789	4839	4889	4939	4989	5039	5089
10	5153	5203	5253	5303	5353	5403	5453	5503	5553	5603

Per determinare AC e disp per uno specifico numero di microcicli (m) si usano le seguenti equazioni:

disp = { (m-13)/514} con arrotondamento

AC = { (m-13) − 514 (disp) }/2

Utilizzando queste equazioni, il tempo di delay sarà il numero esatto od un microciclo meno del numero di microcicli richiesti.

Appendice B

DATA SHEET DI SC/MP

B.1 INTRODUZIONE

In questa appendice presentiamo i fogli tecnici relativi al microprocessore SC/MP, così come vengono forniti dalla casa costruttrice, la National Semiconductor. Come si può notare, il suddetto microprocessore è presente in due versioni:
ISP-8A/500D (SC/MP) realizzato secondo tecnologia MOS p-channel,
(ISP-8A/600, (SC/MP II), MOS n-channel
Le differenze tra queste due versioni sono (trascurando la tecnologia mediante la quale sono realizzati) così piccole da autorizzare la stesura di questo libro impostato esclusivamente su SC/MP p-channel e di indicare in questa appendice le differenze tra le due versioni.

B.2 DIFFERENZE TRA SC/MP E SC/MP II

B.2.1 Differenze elettriche

SC/MP II necessita di una sola tensione di alimentazione, cioè di +5V; inoltre la sua dissipazione è notevolmente inferiore (meno di 250 mW). Entrambe le versioni possono essere considerate TTL compatibili, ma è comunque opportuno notare che esistono delle lievi differenze riguardo le caratteristiche elettriche di ingresso e di uscita: si veda a tale proposito la pag. 2 dei rispettivi data sheet.

B.2.2 Timing

Si può dire che non esistono sostanziali differenze tra le due versioni per quanto riguarda le relazioni di tempo tra i vari segnali; è necessario notare solamente due particolarità:
SC/MP II può operare ad una velocità di clock 4 volte superiore rispetto a SC/MP, inoltre in SC/MP II è possibile utilizzare esternamente il segnale di clock generato dall'oscillatore interno (tramite il pin X out).
Esistono invece delle differenze sostanziali per quanto riguarda il modo di contenere il quarzo o l'eventuale capacità dell'oscillatore esterno, infatti SC/MP II richiede la presenza di un filtro in serie al quarzo, oppure, nel caso in cui quest'ultimo non sia utilizzato, la frequenza di oscillazione è determinata dalla costante di tempo del filtro stesso. Si legga a tale proposito pag. 7 dei rispettivi manuali.

B.2.3. Differenze logiche

Non esiste alcuna differenza logica tra SC/MP e SC/MP II. Entrambe le versioni eseguono lo stesso set di istruzioni, possiedono la medesima struttura interna e la stessa logica di Input/Output. Il tempo di esecuzione delle istruzioni è invece diverso, in quanto SC/MP II è in grado di eseguirle in metà tempo rispetto a SC/MP, se viene fatto operare ad una frequenza di clock di 4 MHz. È invece importante notare come la logica di accesso al bus di SC/MP II sia completamente invertita rispetto SC/MP; infatti:

NBREQ - Quando SC/MP II richiede l'accesso al bus pone questa linea bassa ("0" logico).

NENIN - SC/MP II accede al bus solamente quando questa linea è bassa ("0" logico).

NENOUT - Quando il bus è libero, ma SC/MP II non vuole accedere ad esso, questa linea viene posta dal processore stesso bassa, cioè allo "0" logico.

A questo proposito, si legga il paragrafo 2.3.1 per quanto riguarda la logica di accesso al bus di SC/MP. Successivamente per un confronto immediato tra le due versioni di SC/MP a questo proposito, si vedano i relativi data sheet, e più precisamente:
— fig. 10 ed 11 dei fogli tecnici di SC/MP;
— fig. 11 e 12 dei fogli tecnici di SC/MP II.

NATIONAL

JULY 1976
Pub. No. 420305227-001A

ISP-8A/500D single-chip 8-bit microprocessor (SC/MP)

general description

SC/MP (Simple Cost-effective MicroProcessor) is a single-chip 8-bit microprocessor packaged in a standard, 40-pin, dual-in-line package.

Silicon gate, depletion mode standard-process technology ensures high performance, high reliability, and high producibility.

SC/MP is intended for use in general-purpose applications where cost per function is a most significant criterion. But cost efficiency is only a part of SC/MP's story. It goes on to include a variety of useful functions that are not even provided by some of the expensive microprocessors, like self-contained timing circuitry, 16-bit (65k) addressing capability, serial or parallel data-transfer capability and common memory/peripheral instructions. The built-in features in conjunction with the low initial cost describe what SC/MP really is — a microprocessor specifically designed to provide the simplest and most efficient solution to many application requirements.

features

- Simpler interfacing
  - Bidirectional TRI-STATE® 8-bit data bus
  - TTL-compatible input/output interface
- Direct Memory Access (DMA) and multiprocessor capabilities
  - Handshake bus-access control
- Simplified programming
  - Multiple addressing modes — program-counter-relative, immediate data, indexed, auto-indexed, and implied
- Direct control output
  - Three user-accessible control-flag outputs
- Simpler I/O hardware
  - Separate serial-data input and output ports
  - Two sense inputs
  - Direct interfacing to standard memory parts
- On-chip timing
  - Minimizes external hardware required for clock generation
- Interface flexibility
  - Capability to interface with memories or peripherals of any speed
- Large system capability
  - Address capability to 65k bytes of memory
- Simplified power requirements

block and connection diagrams

TOP VIEW

DA-B30M76/Printed in U.S.A.

applications

- Test Systems and Instrumentation
- Machine Tool Control
- Small Business Machines
- Word Processing Systems
- Educational Systems
- Multiprocessor Systems
- Process Controllers
- Terminals
- Traffic Controls
- Laboratory Controllers
- Sophisticated Games
- Automotive

absolute maximum ratings

Voltage at Any Pin	V_{SS} + 0.3V to V_{SS} – 20V
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	–65°C to +150°C
Lead Temperature (Soldering, 10 seconds)	300°C

electrical characteristics (T_A = 0°C to +70°C, V_{SS} = +5V ± 5%, V_{GG} = –7V ± 5%)

Parameter	Conditions	Min.	Typ.*	Max.	Units
INPUT SPECIFICATIONS					
ENIN, NHOLD, NRST, SENSE A, SENSE B, SIN, DB0-DB7 (TTL Compatible) (Note 2)					
Logic "1" Input Voltage		V_{SS} – 1		V_{SS} + 0.3	V
Logic "0" Input Voltage		V_{SS} – 10		0.8	V
Pullup Transistor "ON" Resistance (Note 2)	V_{IN} = (V_{SS} – 1) V		7.5	12	kΩ
Logic "0" Input Current	V_{IN} = 0V			–1.6	mA
BREQ (Note 3)					
Logic "1" Input Voltage		V_{SS} – 1		V_{SS} + 0.3	V
Logic "0" Input Voltage				0.8	V
X1, X2 (Note 4)					
Logic "1" Input Voltage		3.0		V_{SS} + 0.3	V
Logic "0" Input Voltage				0.4	V
Logic "1" Input Current	V_{IN} = 3.0V			5.0	mA
Logic "0" Input Current	V_{IN} = 0.4V	–5.5			mA
Input Capacitance (All pins except V_{GG} and V_{SS})				10	pF
Supply Current					
I_{GG} (See Typical Plot of Normalized I_{GG} [and I_{SS}] Versus Ambient Temperature on page 6.)	T_A = 0°C, loads on all outputs: I_{SINK} = 1.6mA (See diagram, Simulated Current Load, on page 6.)		100	135	mA
I_{SS}			90	125	mA
OUTPUT SPECIFICATIONS					
BREQ (Note 3)					
Logic "1" Output Current	V_{OUT} = (V_{SS} – 1) V	–2.0			mA
Logic "0" Output Current	$V_{GG} \leq V_{OUT} \leq V_{SS}$			±10	μA
External Load Capacitance				50	pF
All Other Outputs					
Logic "1" Output Voltage	I_{OUT} = –80μA	V_{SS} – 1			V
	I_{OUT} = –200μA	2.4			V
Logic "0" Output Voltage	I_{OUT} = 1.6mA			0.4	V
Logic "0" Output Current	V_{OUT} = –0.5V			4.0	mA
Logic "0" Output Voltage	I_{OUT} = 0mA (unloaded)	–3.0	–0.7		V

*Typical parameters correspond to nominal supply voltage at 25°C.

electrical characteristics (T_A = 0°C to +70°C, V_{SS} = +5V ± 5%, V_{GG} = −7V ± 5%) (continued)

Parameter	Conditions	Min.	Typ.*	Max.	Units
TIMING SPECIFICATIONS (Note 5)					
T_X (Notes 4 and 6)		1.0		10.0	μs
	820pF ± 10% across X1 & X2	1.0		4.0	μs
f_{res}	crystal with equivalent series resistance ⩽ 600Ω	900		1000	kHz
Address and Input/Output Status (See figures 5 and 6.)					
T_{D1} (ADS)		$(3T_X/2) - 150$	$3T_X/2$	$(3T_X/2) + 200$	ns
T_W (ADS)		$(T_X/2) - 250$			ns
T_S (ADDR)		$(T_X/2) - 300$			ns
T_H (ADDR)		30	50		ns
T_S (STAT)		$(T_X/2) - 300$			ns
T_H (STAT)		30	50		ns
Data Input Cycle (See figure 5.)					
T_D (RDS)		−80	−50		ns
T_W (RDS)		$(3T_X/2) - 400$			ns
T_S (RD)		300			ns
T_H (RD)		0			ns
T_{ACC} (RD)		$2T_X - 400$			ns
Data Output Cycle (See figure 6.)					
T_D (WDS)		$T_X - 250$			ns
T_W (WDS)		$T_X - 250$			ns
T_S (WD)		$(T_X/2) - 300$			ns
T_H (WD)		60	100		ns
Input/Output Cycle Extend (See figure 7.)					
T_S (HOLD)		300			ns
T_{D1} (HOLD)				300	ns
T_{D2} (HOLD)				500	ns
T_W (HOLD)				∞	ns
Bus Access (See figure 4.)					
T_D (ENOUT)				300	ns
T_{D2} (ADS)		$(T_X/2) - 350$		$T_X + 500$	ns
OUTPUT LOAD CAPACITANCE					
External Load Capacitance				75	pF

Note 1: Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under electrical characteristics.

Note 2: Pullup transistors provided on chip for TTL compatibility.

Note 3: BREQ is an input/output signal that requires an external resistor to V_{GG} or ground.

Note 4: X_1 and X_2 are master timing inputs that are normally connected to a 1-megahertz crystal or an external capacitor to control the frequency of the on-chip oscillator.

A hermetically sealed quartz crystal is recommended. The crystal must be a series-resonant type and its equivalent series resistance must not exceed 600 ohms. Suppression of third harmonic oscillations may be required depending on the characteristics of the crystal. Typically, a 500-picofarad capacitor across pin X_1 or X_2 and an AC ground minimizes third harmonic effects.

If use of an external oscillator is desired, the circuit shown in figure 3 or an equivalent may be used.

Note 5: All times measured from valid Logic "0" or Logic "1" level.

Note 6: T_X is the time period for one clock cycle of the on-chip or external oscillator. Refer to paragraph titled Timing Control for detailed definition.

*Typical parameters correspond to nominal supply voltage at 25°C.

functional description

SC/MP is a self-contained general-purpose microprocessor designed for ease of implementation in stand-alone, DMA (Direct Memory Access), and multiprocessor applications. Communications between SC/MP and external memory/ peripheral devices are effected via a 12-bit dedicated address bus and an 8-bit bidirectional data bus. During the address interval of each input/output cycle, SC/MP employs both busses to provide a 16-bit address output: the 12 least significant address bits are sent out over the 12-bit address bus and the 4 most significant address bits are sent out over the 8-bit data bus along with 4 status bits. Separate strobe outputs from SC/MP (NADS, NWDS, NRDS) indicate when valid address information is present on the two busses, and when valid input/ output memory or peripheral data are present on the 8-bit bus. To further extend flexibility of application, serial data input/output ports are also provided so that serial data transfers can be effected under program control. The remaining input/output signals shown in figure 1 are dedicated to general-purpose control and status functions, including initialization, bus management, microprocessor halt, interrupt request, input/ output cycle extension, and user-specified hardware/ software interface functions. A detailed description of each input/output signal is provided in table 1.

FIGURE 1. SC/MP Detailed Block Diagram

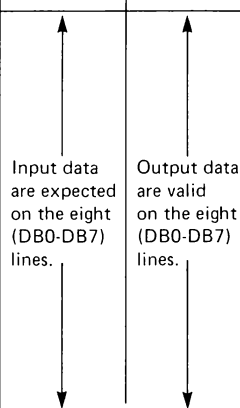



TABLE 1. Input/Output Signal Description

Signal Mnemonic	Functional Name	Description
NRST	Reset Input	Set high for normal operation. When set low, aborts in-process operations. When returned high, internal control circuit zeroes all programmer-accessible registers; then, first instruction is fetched from memory location 0001_{16}.
CONT	Continue Input	When set high, enables normal execution of program stored in external memory. When set low, SC/MP operation is suspended (after completion of current instruction) without loss of internal status.
BREQ	Bus Request Input/Output	Associated with SC/MP internal allocation logic for system bus. Can be used as bus request output or bus busy input. Requires external load resistor to V_{GG}.
ENIN	Enable Input	Associated with SC/MP internal allocation logic for system bus. When set high, SC/MP is granted access to system busses. When set low, places system busses in high-impedance (TRI-STATE®) mode.
ENOUT	Enable Output	Associated with SC/MP internal allocation logic for system bus. Set high when ENIN is high and SC/MP is not using system busses (BREQ-low). Set low at all other times.
NADS	Address Strobe Output	Active-low strobe. While low, indicates that valid address and status output are present on system busses.
NRDS	Read Strobe Output	Active-low strobe. On trailing edge, data are input to SC/MP from 8-bit bidirectional data bus. High-impedance (TRI-STATE®) output when input/output cycle is not in progress.
NWDS	Write Strobe Output	Active-low strobe. While low, indicates that valid output data are present on 8-bit bidirectional data bus. High-impedance (TRI-STATE®) output when input/output cycle not in progress.
NHOLD	Input/Output Cycle Extend Input	When set low prior to trailing edge of NRDS or NWDS strobe, stretches strobe to extend input/output cycle; that is, strobe is held low until NHOLD signal is returned high.
SENSE A	Sense/Interrupt Request Input	Serves as interrupt request input when SC/MP internal IE (Interrupt Enable) flag is set. When IE flag is reset, serves as user-designated sense condition input. Sense condition testing is effected by copying status register to accumulator.
SENSE B	Sense Input	User-designated sense-condition input. Sense-condition testing is effected by copying status register to accumulator.
SIN	Serial Input to E Register	Under software control, data on this line are right-shifted into E register by execution of SIO instruction.
SOUT	Serial Output from E Register	Under software control, data are right-shifted onto this line from E register by execution of SIO instruction. Each data bit remains latched until execution of next SIO instruction.
FLAGS 0, 1, 2	Flag Outputs	User-designated general-purpose flag outputs of status register. Under program control, flags can be set and reset by copying accumulator to status register.
AD00-AD11	Address Bit 00 through Address Bit 11	Twelve TRI-STATE® address output lines. SC/MP outputs 12 least significant address bits on this bus when NADS strobe is low. Address bits are then held valid until trailing edge of read (NRDS) or write (NWDS) strobes. After trailing edge of NRDS or NWDS strobe, bus is set to high-impedance (TRI-STATE®) mode until next NADS strobe.

NOTE:

The 8-bit bidirectional data bus is set to the high-impedance (TRI-STATE®) mode except when it is actually in use by SC/MP (NADS, NRDS, or NWDS low). During the addressing interval of each input/output cycle (NADS low), SC/MP provides address and status outputs over the bus; during the ensuing data-transfer interval (NRDS or NWDS low), 8-bit input or output data bytes are routed over the bus.

TABLE 1. **Input/Output Signal Description** (Continued)

Signal Mnemonic/ Pin Designation	Output at NADS Time		Input at NRDS Time	Output at NWDS Time
	Functional Name	Description		
DB0	Address Bit 12	Fourth most significant bit of 16-bit address.	↑	↑
DB1	Address Bit 13	Third most significant bit of 16-bit address.		
DB2	Address Bit 14	Second most significant bit of 16-bit address.		
DB3	Address Bit 15	Most significant bit of 16-bit address.	Input data are expected on the eight (DB0-DB7) lines.	Output data are valid on the eight (DB0-DB7) lines.
DB4	R-Flag	When high, data input cycle is starting; when low, data output cycle is starting.		
DB5	I-Flag	When high, first byte of instruction is being fetched.		
DB6	D-Flag	When high, indicates delay cycle is starting; that is, second byte of DLY instruction is being fetched.	↓	↓
DB7	H-Flag	When high, indicates that Halt Instruction has been executed. (In some system configurations, the H-Flag output is latched and, in conjunction with the CONTinue input, provides a programmed halt.)	**Note:** The DB0 through DB7 (AD12-HFLG) lines are a high-impedance (open circuit) load when SC/MP does not have access to the input/output bus.	

DRIVERS AND RECEIVERS

Equivalent circuits for SC/MP drivers and receivers are shown below. All inputs have static charge protection circuits consisting of an RC filter and voltage clamp. These devices still should be handled with care, as the protection circuits can be destroyed by excessive static charge.

SC/MP Driver and Receiver Equivalent Circuits

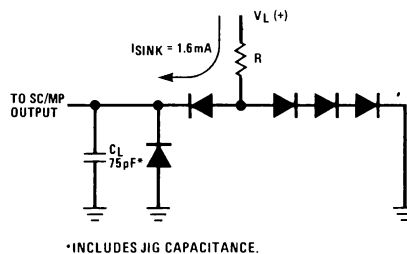
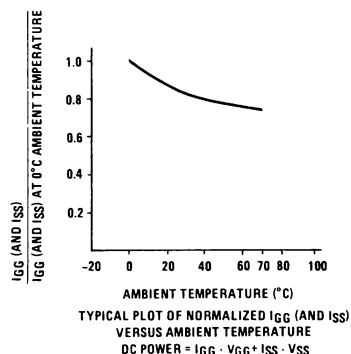
SUPPLY CURRENT DATA

Below are the two diagrams referenced from the parametric specification for the supply current, page 2.

*INCLUDES JIG CAPACITANCE.

Simulated Current Load

TYPICAL PLOT OF NORMALIZED I_{GG} (AND I_{SS}) VERSUS AMBIENT TEMPERATURE

DC POWER = $I_{GG} \cdot V_{GG} + I_{SS} \cdot V_{SS}$

TIMING CONTROL

All necessary timing signals are provided by an on-chip oscillator and a timing generator. The frequency of the oscillator, in turn, is selected by connecting an external capacitor or crystal between pins 37 and 38 (X1 and X2). When a crystal is used, the resulting frequency of the oscillator is equal to the resonant frequency of the crystal; when a capacitor is used, the frequency of the oscillator varies according to the capacitance value as shown in figure 2.

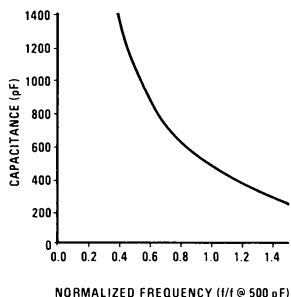



FIGURE 2. Oscillator Frequency versus Capacitance

If desired, the on-chip oscillator can be disabled and the timing generator can be driven by an externally generated clock. In this case, the clock comprises single-phase true and complement inputs. One input is applied to X1 and the other to X2. (A clock is generated at twice the SC/MP clock frequency and is divided by two by a flip-flop as shown in figure 3.)

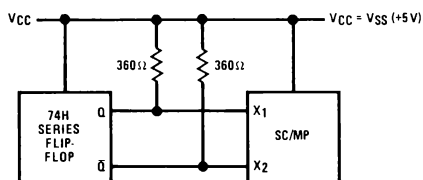



FIGURE 3. External Clock Generation

In the discussions that follow, instruction execution and input/output timing are described in terms of microcycles. For purposes of definition, the time interval of a microcycle is computed according to the following formula:

$$1 \text{ microcycle} = 2T_X$$

where

$$T_X = \text{time period of oscillator} = \frac{1}{f_{osc}} = \frac{1}{f_{res}} = \frac{1}{\text{External Clock Frequency}}$$

f_{osc} = frequency of on-chip oscillator

f_{res} = resonant frequency of quartz crystal connected between pins 37 and 38

INSTRUCTION FORMAT

The SC/MP instruction repertoire includes both single-byte and double-byte instructions. A single-byte instruction consists of an 8-bit operation code that specifies an operation that SC/MP can execute without further reference to memory. A double-byte instruction consists of an 8-bit operation code and an 8-bit data or displacement field. When the second byte represents a data field, the data are processed by SC/MP during execution of the instruction, thereby eliminating the need for further memory references. When the second byte represents a displacement value, it is used to calculate a memory address that will be accessed (written into or read from) during execution of the instruction (refer to Addressing).

DATA STORAGE

As shown in figure 1, SC/MP provides ten internal registers, seven of which are accessible to the programmer. The purpose and function of these registers are described below.

Program Counter — The program counter is a 16-bit register that contains the address of the instruction being executed. The contents of this register are automatically incremented by one just before each instruction is fetched from memory to enable sequential execution of the stored instruction. Under program control, the contents of this register also may be modified or exchanged with the contents of a pointer register to effect subroutine calls and program branches.

NOTE:

The 16-bit address output of the program counter consists of a 4-bit high-order address and a 12-bit low-order address. When the program counter is incremented at the start of each instruction fetch input/output cycle, only the 12 low-order bits are affected; no carry is provided to the 4 high-order bits. For systems employing memories of 4k or less, the high-order bits can be ignored as they are set to 0000_{16} following initialization. For systems employing larger memories, the contents of a pointer register can be modified to select the desired 4k block of memory.

Pointer Registers — The pointer registers are 16-bit general-purpose registers that are loaded normally under program control with reference addresses that serve as page pointers, stack pointers, and subroutine pointers. In applications having minimal memory addressing requirements, these registers may be used alternately as data storage registers.

NOTE:

When interrupt requests are enabled, pointer register 3 is automatically referenced by the internal microprogram for formation of the starting address of the user-generated interrupt service routine. (See figure 8.) In this case, the contents of pointer register 3 must be set to one less than the memory location of the first instruction in the interrupt service routine.

Accumulator — The 8-bit accumulator (AC) is the primary working register of SC/MP. It is used for performing and storing the results of arithmetic and logic operations as well as for data transfers, shifts, rotates, and data exchanges with the program counter, the pointer registers, and the status register.

Extension Register — The extension register is used both for serial input/output data transfers and with the accumulator to effect arithmetic, logic, and data-transfer operations. If the second byte of an indexed or auto-indexed memory-reference instruction (refer to Addressing) equals -128_{10}, the contents of the extension register are used as the displacement value for address formation.

Status Register — The status register provides storage for arithmetic, control, and software status flags. For more-detailed information on the function of this register, refer to Status Register under the description of the Arithmetic and Logic Unit.

Instruction Register — The 8-bit instruction register is not accessible to the programmer. During the fetch phase of each instruction cycle, this register is loaded with the 8-bit instruction operation code retrieved from memory (for a single-byte instruction or the first byte of a double-byte instruction).

Data Input/Output Register — The data input/output register is not accessible to the programmer. It is used for temporary storage of all input/output data received via or transmitted over the 8-bit bidirectional data bus during the data-transfer interval of each input/output cycle (NRDS or NWDS low).

Address Register — The 16-bit address register is not accessible to the programmer. It is used for temporary storage of the 16-bit address transmitted during an input/output cycle.

ARITHMETIC AND LOGIC UNIT

The Arithmetic and Logic Unit (ALU) provides the data-manipulation capability that is an essential feature of any microprocessor. The operations provided by the ALU include OR, XOR, increment, decrement, binary addition, and decimal addition. For decimal addition, the data inputs to the ALU are treated as two 4-bit BCD digits, thereby eliminating the program-storage and execution time required to perform BCD to binary conversion.

BUS TRANSFER LOGIC

The bus transfer logic processes the gating and function control outputs of the instruction-decode logic to provide the shift-right (with link, without link, or with serial input data), rotate (with or without link), and bus-exchange functions necessary for data movement between the SC/MP internal read and write busses. A general summary of the data-manipulation capabilities available to the programmer follows.

1. Either the low-order or the high-order byte of any pointer register can be exchanged with the contents of the 8-bit accumulator. Thus, data exchanges between the pointer registers can be effected one byte at a time via the accumulator.

2. The contents of the program counter can be directly exchanged with the contents of any pointer register.

3. The contents of the extension register can be loaded into the accumulator or can be exchanged with the contents of the accumulator. When the accumulator is loaded from the extension register, the original contents of the accumulator are lost.

4. The contents of the status register can be copied into the accumulator to enable status modification or conditional-branch testing. When the status register is copied into the accumulator, the contents of the status register are not altered but the original contents of the accumulator are lost.

5. The contents of the accumulator can be copied into the status register to change the outputs of the status register, except for status bits 4 and 5 (Sense A and B inputs to SC/MP). Since these are read-only bits, they are not affected by data movements internal to SC/MP. Copying the accumulator into the status register does not alter the contents of the accumulator.

NOTE:

The flag 0, 1, and 2 outputs of the status register serve as latched flags; in other words, they are set to the specified state when the contents of the accumulator are copied into the status register, and they remain in the specified state until the contents of the status register are modified again under program control.

STATUS REGISTER

The function of each bit in the status register is described briefly below.

7	6	5	4	3	2	1	0
CY/L	OV	S_B	S_A	IE	F_2	F_1	F_0

User Flag 0 — User-assigned general-purpose status bit for implementation as software status bit or in system control applications. This status bit is available as an external output from SC/MP.

User Flag 1 — Same as User Flag 0.

User Flag 2 — Same as User Flag 0.

Interrupt Enable Flag — Internal status bit that is set and reset under program control. When set, SC/MP recognizes external interrupt requests received via Sense A input. When reset, inhibits SC/MP from recognizing interrupt requests.

Sense A — General-purpose status input for sensing external conditions. When IE flag is reset, this bit can be tested by copying status register to accumulator. When IE flag is set, this bit serves as interrupt request input causing SC/MP to automatically branch to user-generated interrupt-service routine in response to high input.

Sense B — Same as Sense A except that it is not tested for interrupt status.

NOTE:

Sense A and B inputs are read-only bits. Thus, they are not affected when the contents of the accumulator are copied into the status register.

Overflow (OV) — This bit is set if an arithmetic overflow occurs during an add (ADD, ADI, or ADE) or a complement-and-add instruction (CAD, CAI, or CAE). It is not affected by the decimal-add instructions (DAD, DAI, or DAE).

Carry/Link (CY/L) — This bit is set if a carry from the most significant bit occurs during an add, complement-and-add, or decimal-add instruction. Thus, it serves as a carry input to the next add instruction. In addition, it is included in the Shift Right with Link (SRL) and Rotate Right with Link (RRL) instructions.

CONTROL

The operation of the SC/MP microprocessor consists of repeatedly accessing or fetching instructions from the program stored in external memory and executing the operations specified by the instructions. These two steps are carried out under the control of an internal microprogram. (SC/MP is not user-microprogrammable.) The microprogram is similar to a state table specifying the series of states of system control signals necessary to carry out each instruction. Microprogram storage is provided in the instruction decode and control logic, and microprogram routines are implemented to fetch and execute instructions. The fetch routine first increments the program counter, and then causes the instruction address to be transferred from the program counter to the system busses via the output address register. The microprogram next initiates an input data transfer. When the instruction operation code is subsequently placed on the 8-bit data bus (single-byte instruction or first byte of double-byte instruction), the operation code is loaded into the instruction register. The operation code is then partially decoded to determine whether the instruction contains a second byte. If it does, a second input data transfer is effected to load the next byte in the data input/output register.

After the complete instruction is stored in the instruction and/or data input/output register(s), the instruction decoder transforms the instruction operation code into the address of the appropriate instruction-execution routine contained in the internal microprogram. The microprogram then branches to the specified internal address to initiate execution of the instruction. The resulting execution routine comprises one or more microinstructions that implement the required functions. For example, the first microcycle of an Extension Register Add Instruction (ADE) causes the contents of the extension register to be gated onto the read bus, transferred to the write bus via the bus control logic, and then written into the data input/output register. The next microcycle causes the contents of the accumulator to be gated onto the read bus, the contents of the read bus to be added to the contents of the data input/output register via the ALU, and the resultant output of the ALU to be written into the accumulator via the write bus. The final step of the execution routine is a jump back to the fetch routine to access the next instruction.

INITIALIZATION

Since SC/MP may power up in a random condition, the following power-up and initialization procedure is recommended.

1. Apply power (V_{SS} and V_{GG}) and set NRST low.

   NOTE:

   Allow ample time (typically, 100ms) for the oscillator and the internal clocks to stabilize. In systems where NRST is set low after turning on power, NRST must remain low for a minimum of $4T_X$. While NRST is low, any in-process operations are aborted automatically. When NRST is low, strobes and address and data busses are in the Non-I/O state (high-Z state).

2. Set NRST high.

   NOTE:

   This causes the SC/MP internal control circuit to set the contents of all programmer-accessible registers to zero. Thus, when SC/MP is granted access to the system busses following initialization, the first instruction is fetched *always* from memory location 0001_{16}. The BREQ output goes high, indicating the start of this input/output cycle; this occurs at a time within $13T_X$ after NRST is set high. Normal execution of the program continues as long as NRST remains high.

parallel data transfers

Parallel data transfers occur during each instruction fetch and during the ensuing read/write cycle associated with execution of the memory-reference instructions. This class of instruction could perhaps more properly be called the "Input/Output Reference Class" in the case of the SC/MP microprocessor, since all data transfers, whether with memory, peripheral devices, or a central processor data bus, occur through the execution of these instructions. This unified bus structure is in contrast with many other microprocessors and minicomputers that have one instruction type (input/output class) for communication with peripheral devices and another instruction type (memory reference class) for communication with memories. The advantage of the approach taken by SC/MP is that a wider variety of instructions (the entire memory-reference class) is available for communications with peripherals. Thus, the LD and ST (Load and Store) instructions can be used for basic transfers, the ILD and DLD (increment/decrement and load) instructions can be used for indexing peripheral registers, and the remaining memory reference instructions can be used, as required, for "one-step" retrieval and processing of peripheral input data.

BUS ACCESS

Before SC/MP can initiate parallel data transfers with memory or peripheral devices, it must have access to the system address and data busses. Three of the SC/MP input/output signals are associated with bus control: BREQ, ENIN, and ENOUT. For simple stand-alone applications, the ENOUT signal can be ignored and the ENIN signal can be tied to V_{SS} to allow the SC/MP microprocessor to have continual access to the system busses. The BREQ input/output line then goes high during each input/output cycle as shown in figures 5 and 6 to indicate when SC/MP is actually using the system busses.

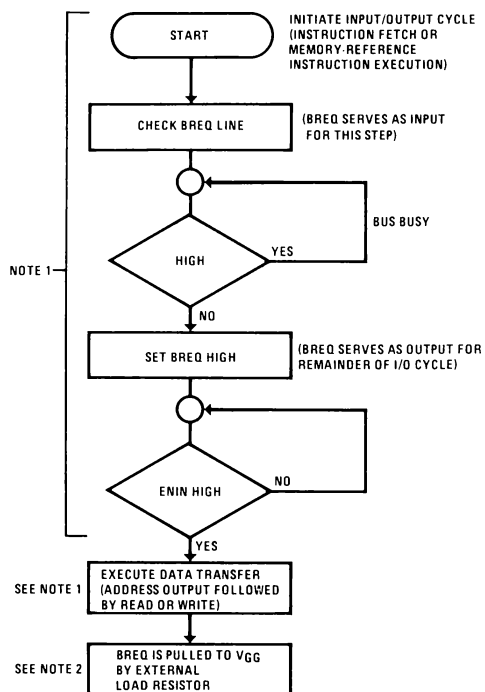
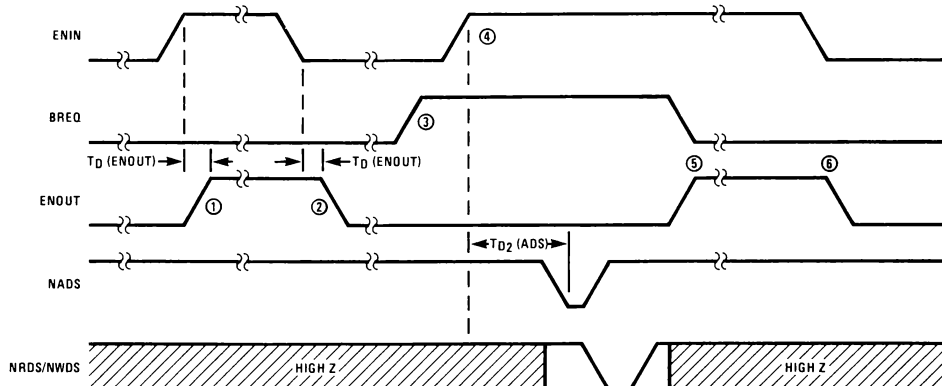
NOTE:

The BREQ input/output line must be tied to V_{GG} (or ground in +5-volt/-7-volt systems) via an external load resistor to allow normal operation of the SC/MP microprocessor.

For DMA and multiprocessor applications, the BREQ, ENIN, and ENOUT signals can be interconnected in various configurations to allow bus access to be granted to requesting devices according to user-specified priorities. Figure 4 illustrates the general sequence in which these signals are processed by SC/MP to gain access to the system busses and to indicate when the busses are actually being used.

A. BREQ AND ENIN PROCESSING SEQUENCE

Note 1: ENOUT is always low while SC/MP is actually using bus; that is, ENIN input and BREQ output are high.

Note 2: When SC/MP is not using bus (BREQ output or ENIN input low), ENOUT is held in same state as ENIN input.

B. BREQ, ENIN, and ENOUT Timing

Note 1: ENOUT goes high to indicate that SC/MP was granted access to bus (ENIN high) but is not using bus.

Note 2: ENOUT goes low in response to low ENIN input.

Note 3: SC/MP generates bus request; bus access not granted because ENIN low.

Note 4: ENIN goes high. Bus access now granted and input/output cycle actually initiated. If ENIN is set low while SC/MP has access to the bus, the address and data ports will go to the high-impedance (TRI-STATE®) state, but BREQ will remain high. When ENIN is subsequently set high, the input/output cycle will begin again.

Note 5: I/O cycle completed. ENOUT goes high to indicate that SC/MP granted access to bus but not using bus. If ENIN had been set low before completion of input/output cycle, ENOUT would have remained low.

Note 6: ENOUT goes low to indicate that system busses are available for use by highest-priority requestor.

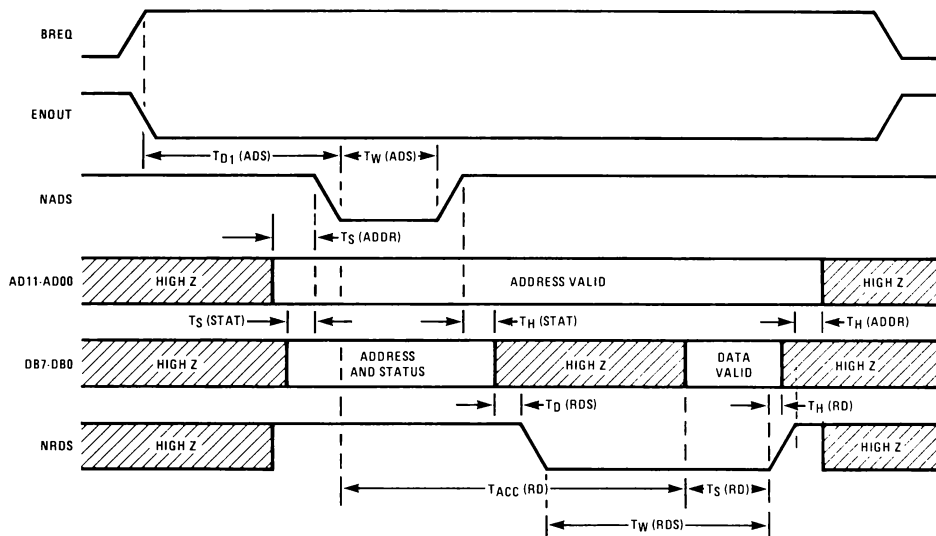
FIGURE 4. Bus Access Control

Note: Timing is valid when ENIN is wired high or is set high before BREQ is set high by SC/MP; see figure 4 for NADS timing when ENIN is set high after BREQ.

FIGURE 5. SC/MP Data Input Timing

BREQ
ENOUT
T_{D1} (ADS)
T_W (ADS)
NADS
T_S (ADDR)
AD11-AD00
HIGH Z
ADDRESS VALID
HIGH Z
T_S (STAT)
T_H (STAT)
T_H (ADDR)
DB7-DB0
HIGH Z
ADDRESS AND STATUS
WRITE DATA VALID
HIGH Z
T_S (WD)
T_H (WD)
NWDS
HIGH Z
HIGH Z
T_D (WDS)
T_W (WDS)

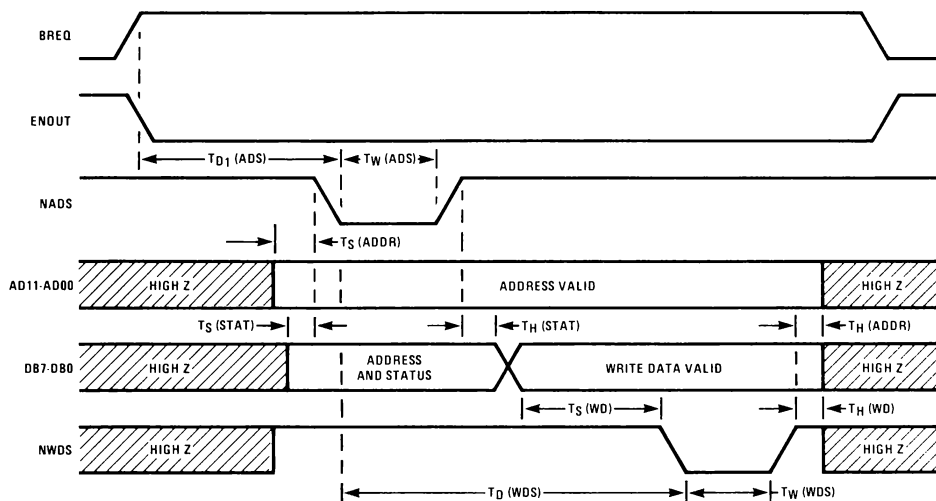



FIGURE 6. SC/MP Data Output Timing

INPUT/OUTPUT CYCLE

Once SC/MP has control of the system busses, the actual input/output cycle begins. As shown in figures 5 and 6, the functions of memory addressing, data reading, and data writing are implemented, respectively, by the address strobe (NADS), the read strobe (NRDS), and the write strobe (NWDS). Note that the BREQ signal is reset low at the end of the input/output cycle to indicate that the system busses are now free for use by the highest-priority requesting device.

The first operation that SC/MP performs for each input/output cycle is to load the 12 least significant address bits onto the 12-bit address bus, and the 4 most significant address bits along with 4 status bits onto the 8-bit data bus. At the same time, SC/MP sets the NADS output low to indicate that the address and the status information are valid. The low-order address on the 12-bit bus is then held valid for the duration of the input/output cycle; the high-order address and the status information on the 8-bit bus remain valid only while NADS is low. While valid, the status bits have the following significance:

RFLG – When high, indicates that input/output cycle is read cycle; when low, indicates that input/output cycle is write cycle.

IFLG – Set high to indicate that instruction operation code (single-byte instruction or first byte of double-byte instruction) will be output from memory following NADS.

DFLG – Set high only when second byte of Delay Instruction is to be read from memory following NADS. Execution of the Delay Instruction then starts at trailing edge of NRDS. Upon completion, SC/MP provides NADS output to initiate next input/output cycle if bus access is granted. Time in microcycles from leading edge of delay flag to leading edge of subsequent NADS output is computed from the following formula:

Delay = $[9 + 2(AC) + 2\ \text{disp} + 2^9\ \text{disp}]$ microcycles

where

(AC) = unsigned contents of accumulator

disp = unsigned displacement value contained in second byte of Delay Instruction

The time derived from the above formula does not include the four microcycles required to fetch the first byte of the Delay Instruction. Thus, when the Delay Instruction is used for software timing, total instruction execution time equals $[13 + 2(AC) + 2\ \text{disp} + 2^9\ \text{disp}]$ microcycles.

NOTE:

When Halt Instruction is executed, instruction decode and control logic inhibits incrementing of program counter for one input/output cycle. Thus, Halt Instruction is read from memory a second time to enable generation of HFLG output, but no further processing of Halt Instruction occurs. In effect, this procedure ensures HFLG is output in advance of the next instruction to be fetched from memory.

HFLG – Set high only during addressing interval of read cycle that follows Halt Instruction. HFLG may be used to cause user-provided external logic to set the CONT input low, and thereby to effect a programmed halt. Since HFLG read cycle precedes the next instruction fetch, termination of programmed halt enables fetch of first instruction that follows Halt Instruction.

After resetting the NADS output, SC/MP generates an NRDS or NWDS strobe, respectively, to initiate a data-input (read) or data-output (write) operation. For a read operation, input data are strobed into SC/MP from the 8-bit bus on the trailing edge of the NRDS strobe. For a write operation, SC/MP places valid output data on the 8-bit bus on the leading edge of the NWDS strobe. After resetting the NRDS or NWDS strobe to complete the data transfer, SC/MP then resets the BREQ signal to indicate that the system busses are free for use by another controller.

input/output cycle extension

For systems employing memories or peripherals with long access times it may be desirable to utilize the NHOLD input to lengthen the input/output cycle. As shown in figure 7, setting the NHOLD signal low prior to the trailing edge of the NRDS or NWDS strobe causes SC/MP to hold the strobe active until after the NHOLD signal is returned high.

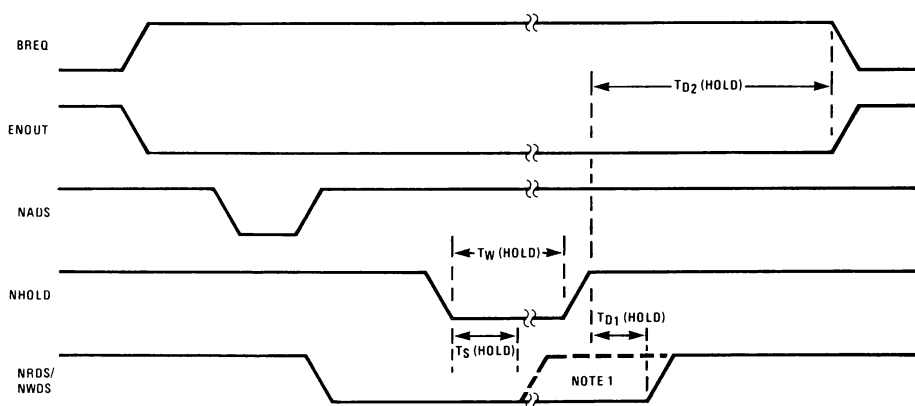
The NHOLD signal can also be used for single-cycle execution of the operating program as required for debugging software.

Note: Dashed trailing edge of NRDS/NWDS indicates normal strobe timing when NHOLD is not active.

FIGURE 7. Extended Input/Output Timing

serial data transfers

Serial input/output data transfers can be used efficiently with very slow input/output peripherals such as X-Y plotters, teletypewriters, slow-speed printers, and so forth. Such transfers can be effected in any of the following manners:

1. By assigning serial input/output functions to the extension register via the SIO (Serial Input/Output) Instruction. When this instruction is executed, the contents of the extension register are shifted right one bit. At the same time, data present on the SIN line are shifted into bit position 7 of the extension register and the original contents of bit position 0 are shifted into a flip-flop to provide a latched output of the SOUT line. The SOUT data are then held latched until the next SIO instruction is executed.

2. By using one of the status flags as an output data bit and one of the sense lines as an input data bit.

3. By implementing external logic such that only one line of the 8-bit data input/output bus is used.

For synchronous systems, serial data input/output timing may be provided by program loops that employ the delay instruction, or by using one or more of the transfer instructions (see table 2) to test the output of an external timing circuit. For asynchronous systems, one of the sense inputs can be used for testing bit-received/ready status and a pulsed flag output can be provided, under program control, for peripheral indexing each time that a data bit is actually shifted in or out.

Systems that have several input/output devices must be multiplexed; device selection can then be accomplished using the status flag outputs of SC/MP, or by using parallel input/output commands to load an external latch. Systems that do not require serial input/output capability can employ the SIN and SOUT lines as a sense input and flag output, respectively.

interrupts

When the internal interrupt enable (IE) flag is set under program control, the Sense A line is enabled to serve as an interrupt request input; when the IE flag is reset, SC/MP is inhibited from detecting interrupts. Thus, while the IE flag is set, the Sense A input is tested prior to the fetch phase of each instruction as shown in figure 8. Upon detection of an interrupt request (Sense A high), the following events occur automatically.

1. The status register IE flag is reset to prevent SC/MP from responding to any further interrupt requests. Interrupt request capability can then be reenabled during or at the end of the ensuing user-generated interrupt service routine via the IEN (Enable Interrupt) Instruction or by copying the accumulator into the status register.

2. The contents of the program counter are exchanged with the contents of the pointer register 3.

3. The contents of the program counter are incremented by one to address the first instruction of the user-generated interrupt service routine.

The interrupt system must be armed before interrupts are enabled. This is accomplished as follows:

1. First, the Interrupt Enable Bit in the Status Register is set true by executing either an Enable Interrupt Instruction (IEN) or a Copy Accumulator to Status Register Instruction (CAS).

2. Second, one additional instruction is fetched and executed.

A return from interrupt is accomplished by executing two instructions: Enable Interrupt (IEN) immediately followed by Exchange Pointer 3 with Program Counter (XPPC 3).

microprocessor halt

The CONT input to SC/MP is provided to enable suspension of operation without loss of internal status. Processing of the CONT input is shown in figure 8. Since this is an asynchronous input, it can be controlled by external timing logic, or as stated previously, the HALT flag output that appears on the 8-bit data bus (during the read cycle that follows execution of a Halt Instruction) can be used with an external circuit to effect a programmed halt condition. Note that when an interrupt request is detected while the CONT input is low, the first instruction of the user-generated interrupt service routine is automatically executed. Thus, the first instruction of the interrupt service routine can be used to reset the external CONT input logic and, thereby, to terminate the microprocessor halt condition if so desired.

After execution of an instruction, the CONT input must be high for a minimum time of $2T_X$ (1 microcycle) in order to fetch and execute the next instruction.

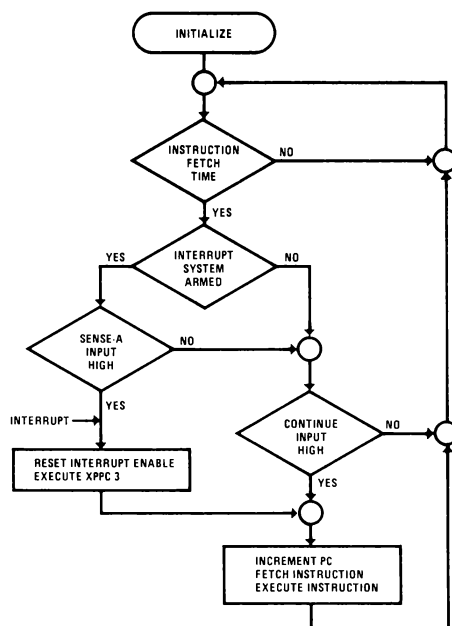



FIGURE 8.

Microprocessor Halt and Interrupt Request Input Processing

instruction set

The SC/MP instruction set provides the general-purpose user of microprocessors a powerful programming capability along with above-average flexibility and speed. The instruction set consists of 46 instructions, which comprise eight general categories. A listing of the complete instruction set is provided in table 2; typical instruction execution times are given in table 3, and notations and symbols used as shorthand expressions of instruction capability are defined in table 4.

ADDRESSING

During execution, instructions and data defined in a program are stored into and loaded from specific memory locations, the accumulator, or selected registers. Because SC/MP, memory (read/write and read-only), and peripherals are on a common data bus, any instruction used to address memory may be used to address the peripherals. The formats of the instruction groups that reference memory are shown below.

7, , , ,3	2	1,0	7, , , , , , ,0	
opcode	m	ptr	disp	Memory Reference Instructions
opcode		ptr	disp	Memory Increment/Decrement Instructions and Transfer Instructions

Memory-reference instructions use the PC-relative, indexed, or auto-indexed methods of addressing memory. The memory-increment/decrement instructions and the transfer instructions use the PC-relative or indexed methods of addressing.

The various methods of addressing memory and peripherals are shown below.

Immediate addressing is an addressing format specific to the immediate instruction group.

Type of Addressing	Operand Formats		
	m	ptr	disp
PC-relative	0	0	-128 to +127
Indexed	0	1, 2, or 3	-128 to +127
Immediate	1	0	-128 to +127
Auto-indexed	1	1, 2, or 3	-128 to +127

For PC-relative, indexed, and auto-indexed memory-reference instructions, another feature of the addressing architecture is that the contents of the extension register are substituted for the displacement if the instruction displacement equals -128 (-X'80).

NOTE:

All arithmetic operations associated with address formation affect only the 12 low-order address bits; no carry is provided to the 4 high-order bits. For systems employing memories of 4k or less, the high-order bits can be ignored as they are set to 0000 following initialization. For systems employing larger memories, the high-order bits must be set to the starting address of the desired 4k block of memory. For example:

0001_2 enables memory locations 1000_{16} - $1FFF_{16}$ to be addressed.

0010_2 enables memory locations 2000_{16} - $2FFF_{16}$ to be addressed and so forth.

PC-Relative Addressing — A PC-relative address is formed by adding the displacement value specified in the operand field of the instruction to the current contents of the program counter. The displacement is an 8-bit twos-complement number, so the range of the PC-relative addressing format is -128_{10} to $+127_{10}$ locations from the current contents of the program counter.

Immediate Addressing — Immediate addressing uses the value in the second byte of a double-byte instruction as the operand for the operation to be performed (see below).

For example, compare a Load (LD) instruction to a Load Immediate (LDI) instruction. The Load instruction uses the contents of the second byte of the instruction in computing the effective address of the data to be loaded. The Load Immediate instruction uses the contents of the second byte as the data to be loaded.

Indexed Addressing — Indexed addressing enables the programmer to address any location in memory through the use of the pointer register and the displacement. When indexed addressing is specified in an instruction, the contents of the designated pointer register are added to the displacement to form the effective address. The contents of the pointer register are not modified by indexed addressing.

Auto-Indexed Addressing — Auto-indexed addressing provides the same capabilities as indexed addressing along with the ability to increment or decrement the designated pointer register by the value of the displacement. If the displacement is less than zero, the pointer register is decremented by the displacement before the contents of the effective address are fetched or stored. If the displacement is equal to or greater than zero, the pointer register is used as the effective address, and the pointer register is incremented by the displacement after the contents of the effective address are fetched or stored.

TABLE 2. SC/MP Instruction Summary

DOUBLE-BYTE INSTRUCTIONS

MNEMONIC	DESCRIPTION	OBJECT FORMAT	OPERATION	MICRO-CYCLES
	Memory Reference Instructions	7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		
LD	Load	1 1 0 0 0 m ptr / disp	(AC)←(EA)	18
ST	Store	1 1 0 0 1	(EA)←(AC)	18
AND	AND	1 1 0 1 0	(AC)←(AC) $\wedge$ (EA)	18
OR	OR	1 1 0 1 1	(AC)←(AC) $\vee$ (EA)	18
XOR	Exclusive-OR	1 1 1 0 0	(AC)←(AC) $\forall$ (EA)	18
DAD	Decimal Add	1 1 1 0 1	(AC)←$(AC)_{10}$ + $(EA)_{10}$ + (CY/L);(CY/L)	23
ADD	Add	1 1 1 1 0	(AC)←(AC) + (EA) + (CY/L);(CY/L),(OV)	19
CAD	Complement and Add	1 1 1 1 1	(AC)←(AC) + ~(EA) + (CY/L);(CY/L),(OV)	20
	Memory Increment/Decrement Instructions	7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		
ILD	Increment and Load	1 0 1 0 1 0 ptr / disp	(AC), (EA)←(EA) + 1	22
DLD	Decrement and Load	1 0 1 1 1 0	(AC), (EA)←(EA) − 1	22
	Immediate Instructions	7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		
LDI	Load Immediate	1 1 0 0 0 1 0 0 / data	(AC)← data	10
ANI	AND Immediate	1 1 0 1 0 1 0 0	(AC)←(AC) $\wedge$ data	10
ORI	OR Immediate	1 1 0 1 1 1 0 0	(AC)←(AC) $\vee$ data	10
XRI	Exclusive-OR Immediate	1 1 1 0 0 1 0 0	(AC)←(AC) $\forall$ data	10
DAI	Decimal Add Immediate	1 1 1 0 1 1 0 0	(AC)←$(AC)_{10}$ + $data_{10}$ + (CY/L);(CY/L)	15
ADI	Add Immediate	1 1 1 1 0 1 0 0	(AC)←(AC) + data + (CY/L);(CY/L),(OV)	11
CAI	Complement and Add Immediate	1 1 1 1 1 1 0 0	(AC)←(AC) + ~data + (CY/L);(CY/L),(OV)	12
	Transfer Instructions	7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		
JMP	Jump	1 0 0 1 0 0 ptr / disp	(PC)←EA	11
JP	Jump if Positive	1 0 0 1 0 1	If (AC) ≥ 0, (PC)←EA	9, 11
JZ	Jump if Zero	1 0 0 1 1 0	If (AC) = 0, (PC)←EA	9, 11
JNZ	Jump if Not Zero	1 0 0 1 1 1	If (AC) ≠ 0, (PC)←EA	9, 11
	Double-Byte Miscellaneous Instructions	7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0		
DLY	Delay	1 0 0 0 1 1 1 1 / disp	count AC to −1, delay = 13 + 2(AC) + 2 disp + 2^9 disp microcycles	13 to 131,593

SINGLE-BYTE INSTRUCTIONS

MNEMONIC	DESCRIPTION	OBJECT FORMAT	OPERATION	MICRO-CYCLES
	Extension Register Instructions	7 6 5 4 3 2 1 0		
LDE	Load AC from Extension	0 1 0 0 0 0 0 0	(AC)←(E)	6
XAE	Exchange AC and Extension	0 0 0 0 0 0 0 1	(AC)↔(E)	7
ANE	AND Extension	0 1 0 1 0 0 0 0	(AC)←(AC) $\wedge$ (E)	6
ORE	OR Extension	0 1 0 1 1 0 0 0	(AC)←(AC) $\vee$ (E)	6
XRE	Exclusive-OR Extension	0 1 1 0 0 0 0 0	(AC)←(AC) $\forall$ (E)	6
DAE	Decimal Add Extension	0 1 1 0 1 0 0 0	(AC)←$(AC)_{10}$ + $(E)_{10}$ + (CY/L);(CY/L)	11
ADE	Add Extension	0 1 1 1 0 0 0 0	(AC)←(AC) + (E) + (CY/L);(CY/L),(OV)	7
CAE	Complement and Add Extension	0 1 1 1 1 0 0 0	(AC)←(AC) + ~(E) + (CY/L);(CY/L),(OV)	8
	Pointer Register Move Instructions	7 6 5 4 3 2 1 0		
XPAL	Exchange Pointer Low	0 0 1 1 0 0 ptr	(AC)↔$(PTR_{7:0})$	8
XPAH	Exchange Pointer High	0 0 1 1 0 1	(AC)↔$(PTR_{15:8})$	8
XPPC	Exchange Pointer with PC	0 0 1 1 1 1	(PC)↔(PTR)	7
	Shift, Rotate, Serial I/O Instructions	7 6 5 4 3 2 1 0		
SIO	Serial Input/Output	0 0 0 1 1 0 0 1	(E_i)→(E_{i-1}), SIN→(E_7), (E_0)→SOUT	5
SR	Shift Right	0 0 0 1 1 1 0 0	(AC_i)→(AC_{i-1}), 0→(AC_7)	5
SRL	Shift Right with Link	0 0 0 1 1 1 0 1	(AC_i)→(AC_{i-1}), (CY/L)→(AC_7)	5
RR	Rotate Right	0 0 0 1 1 1 1 0	(AC_i)→(AC_{i-1}), (AC_0)→(AC_7)	5
RRL	Rotate Right with Link	0 0 0 1 1 1 1 1	(AC_i)→(AC_{i-1}), (AC_0)→(CY/L)→(AC_7)	5
	Single-Byte Miscellaneous Instructions	7 6 5 4 3 2 1 0		
HALT	Halt	0 0 0 0 0 0 0 0	Pulse H-flag	8
CCL	Clear Carry/Link	0 0 0 0 0 0 1 0	(CY/L)←0	5
SCL	Set Carry/Link	0 0 0 0 0 0 1 1	(CY/L)←1	5
DINT	Disable Interrupt	0 0 0 0 0 1 0 0	(IE)←0	6
IEN	Enable Interrupt	0 0 0 0 0 1 0 1	(IE)←1	6
CSA	Copy Status to AC	0 0 0 0 0 1 1 0	(AC)←(SR)	5
CAS	Copy AC to Status	0 0 0 0 0 1 1 1	(SR)←(AC)	6
NOP	No Operation	0 0 0 0 1 0 0 0	None	5

TABLE 3. Instruction Execution Time

INSTRUCTION	READ CYCLES	WRITE CYCLES	TOTAL MICROCYCLES	INSTRUCTION	READ CYCLES	WRITE CYCLES	MICROCYCLES
ADD	3	0	19	JP	2	0	9, 11 for Jump
ADE	1	0	7	JZ	2	0	9, 11 for Jump
ADI	2	0	11	LD	3	0	18
AND	3	0	18	LDE	1	0	6
ANE	1	0	6	LDI	2	0	10
ANI	2	0	10	NOP	1	0	5
CAD	3	0	20	OR	3	0	18
CAE	1	0	8	ORE	1	0	6
CAI	2	0	12	ORI	2	0	10
CAS	1	0	6	RR	1	0	5
CCL	1	0	5	RRL	1	0	5
CSA	1	0	5	SCL	1	0	5
DAD	3	0	23	SIO	1	0	5
DAE	1	0	11	SR	1	0	5
DAI	2	0	15	SRL	1	0	5
DINT	1	0	6	ST	2	1	18
DLD	3	1	22	XAE	1	0	7
DLY	2	0	13 - 131593	XOR	3	0	18
HALT	2	0	8	XPAH	1	0	8
IEN	1	0	6	XPAL	1	0	8
ILD	3	1	22	XPPC	1	0	7
JMP	2	0	11	XRE	1	0	6
JNZ	2	0	9, 11 for Jump	XRI	2	0	10

Note: If slow memory is being used, the appropriate delay should be added for each read or write cycle.

TABLE 4. Symbols and Notations Used to Express Instruction Execution

SYMBOL AND NOTATION	MEANING
AC	8-bit Accumulator.
CY/L	Carry/Link Flag in the Status Register.
data	Signed, 8-bit immediate data field.
disp	Displacement; represents an operand in a nonmemory reference instruction or an address modifier field in a memory reference instruction. It is a signed twos-complement number.
EA	Effective Address as specified by the instruction.
E	Extension Register; provides for temporary storage, variable displacements and separate serial input/output port.
i	Unspecified bit of a register.
IE	Interrupt Enable Flag.
m	Mode bit, used in memory reference instructions. Blank parameter sets m = 0, @ sets m = 1.
OV	Overflow Flag in the Status Register.
PC	Program Counter (Pointer Register 0); during address formation, PC points to the last byte of the instruction being executed.
ptr	Pointer Register (ptr = 0 through 3). The register specified in byte 1 of the instruction.
$ptr_{n:m}$	Pointer register bits; n:m = 7 through 0 or 15 through 8.
SIN	Serial Input pin.
SOUT	Serial Output pin.
SR	8-bit Status Register.
( )	Means "contents of." For example, (EA) is contents of Effective Address.
[ ]	Means optional field in the assembler instruction format.
~	Ones complement of value to right of ~.
→	Means "replaces."
←	Means "is replaced by."
↔	Means "exchange."
@	When used in the operand field of the instruction, sets the mode bit (m) to 1 for auto-incrementing/auto-decrementing indexing.
10^+	Modulo 10 addition.
Λ	AND operation.
V	Inclusive-OR operation.
$\veebar$	Exclusive-OR operation.
⩾	Greater than or equal to.
=	Equals.
≠	Does not equal.

system implementation

Figures 9 through 11 illustrate typical SC/MP system configurations. In figure 9, SC/MP is shown interconnected to three memory devices to form a stand-alone 4-device system that provides 256 woıds of read/write memory and 2,048 words for program storage. Figure 10 shows SC/MP interconnected to an external controller for Direct Memory Access (DMA) operation, and figure 11 illustrates a multiprocessor application using SC/MP's built-in logic to control bus access.

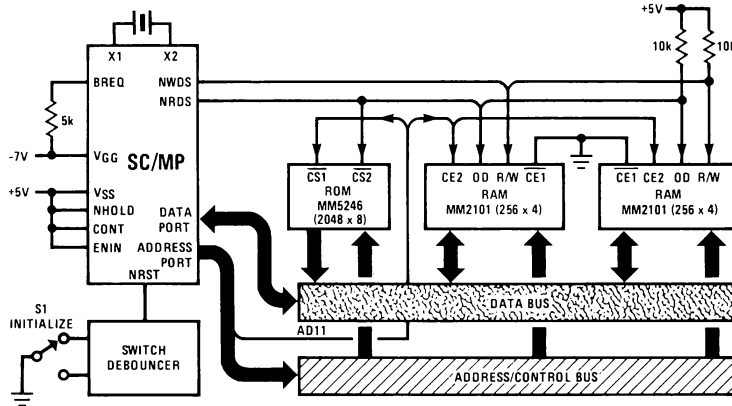



FIGURE 9. SC/MP Four-Chip System

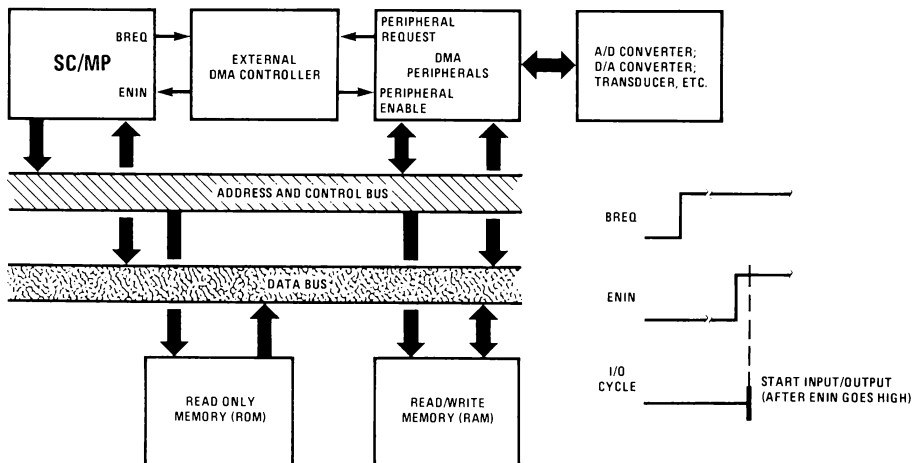



FIGURE 10. SC/MP Interconnected for Direct Memory Access (DMA) Operation

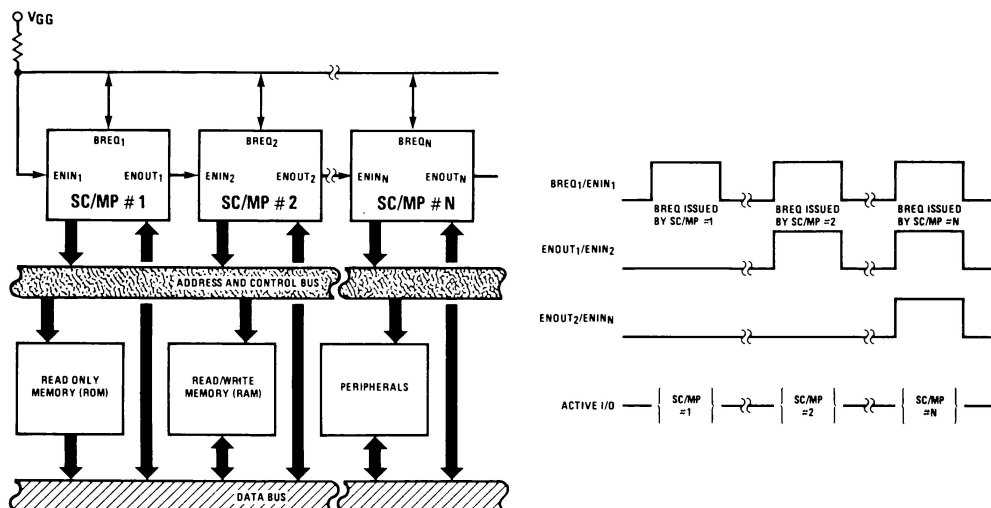
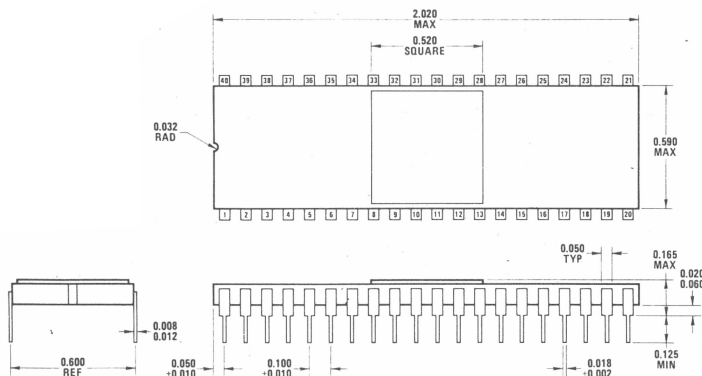



FIGURE 11. Multiprocessor System Using SC/MP Built-in Logic for Bus Control

physical dimensions

ordering information

The SC/MP device may be ordered through the local National Semiconductor sales representative or by contacting our world or international headquarters listed below. The order number is as follows:

ISP-8A/500D

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 732-5000/TWX (910) 339-9240

National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 27649

National Semiconductor (UK) Ltd
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632

NOVEMBER 1976
Pub. No. 426305290-001A
Preliminary

ISP-8A/600 single-chip 8-bit n-channel microprocessor (SC/MP-II)

general description

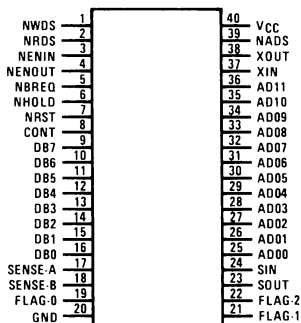
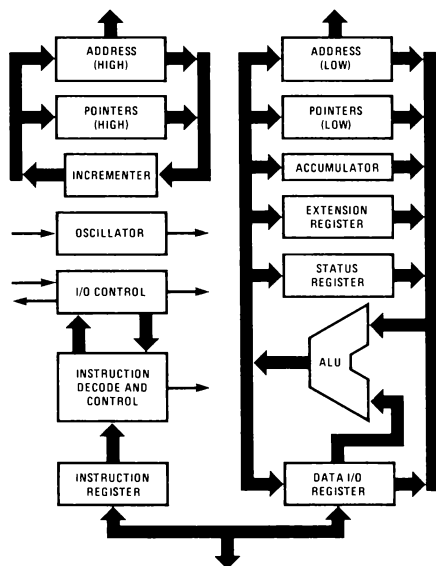
SC/MP (Simple Cost-effective MicroProcessor) is a single-chip 8-bit microprocessor packaged in a standard, 40-pin, dual-in-line package.

N-channel, silicon gate, depletion mode standard-process technology ensures high performance, high reliability, and high producibility.

SC/MP is intended for use in general-purpose applications where cost per function is a most significant criterion. But cost efficiency is only a part of SC/MP's story. It goes on to include a variety of useful functions that are not even provided by some of the expensive microprocessors, like self-contained timing circuitry, 16-bit (65k) addressing capability, serial or parallel data-transfer capability and common memory/peripheral instructions. The built-in features in conjunction with the low initial cost describe what SC/MP really is — a microprocessor specifically designed to provide the simplest and most efficient solution to many application requirements.

customer benefits

- Simpler interfacing
  - Bidirectional TRI-STATE® 8-bit data bus
  - TTL-compatible input/output interface
- Si-gate N-channel ion-implant process
- Direct Memory Access (DMA) and multiprocessor capabilities
  - Handshake bus-access control on chip
- Simplified programming
  - Multiple addressing modes — program-counter-relative, immediate data, indexed, auto-indexed, and implied
- Direct control output
  - Three user-accessible control-flag outputs
- Simpler I/O hardware
  - Separate serial-data input and output ports
  - Two sense inputs
  - Direct interfacing to standard memory parts
- Simplified timing hardware
  - On-chip clock generator
- Interface flexibility
  - Capability to interface with memories or peripherals of any speed
- Large system capability
  - Address capability to 65k bytes of memory
- Simplified power requirements
  - Single 5-volt supply
  - Low power
- Lower cost
  - Plastic package

block and connection diagrams

SC/MP-II Pin Configuration

DA-B20M116/Printed in U.S.A.

applications

- Test Systems and Instrumentation
- Machine Tool Control
- Small Business Machines
- Word Processing Systems
- Educational Systems
- Multiprocessor Systems
- Process Controllers
- Terminals
- Traffic Controls
- Laboratory Controllers
- Sophisticated Games
- Automotive

absolute maximum ratings (Note 1)

Voltage at Any Pin . . . . −0.5V to +7.0V

Operating Temperature Range . . . . 0°C to +70°C

Storage Temperature Range. . . . . . . −65°C to +150°C

Lead Temperature (Soldering, 10 seconds) . . . . . 300°C

dc electrical characteristics (T_A = 0°C to +70°C, V_{CC} = +5V ± 5%)

Parameter	Conditions	Min.	Typ. (Note 2)	Max.	Units
INPUT SPECIFICATIONS					
All Input Pins Except V_{CC} and GND Logic "1" Input Voltage		2.0		V_{CC}	V
Logic "0" Input Voltage		−0.5		0.8	V
Input Capacitance (All pins except V_{CC} and GND)			6		pF
Supply Current I_{CC}	T_A = 25°C outputs unloaded		45		mA
OUTPUT SPECIFICATIONS					
"TRI-STATE®" Pins (NWDS, NRDS, DB0 - DB7, AD00 - AD11) Logic "1" Output Voltage	I_{OUT} = −100μA	2.4			V
Logic "0" Output Voltage	I_{OUT} = 1.6mA			0.4	V
NADS, FLAG 0 - 2, SOUT, NENOUT Logic "1" Output Voltage	I_{OUT} = −100μA	V_{CC} − 1			V
Logic "1" Output Voltage	I_{OUT} = −1mA	1.5			V
Logic "0" Output Voltage	I_{OUT} = 1.6mA			0.4	V
NBREQ (Note 3) Logic "0" Output Voltage	I_{OUT} = 1.6mA			0.4	V
Logic "1" Output Current	$0 \leqslant V_{OUT} \leqslant V_{CC}$			±10	μA
XOUT Logic "1" Output Voltage	I_{OUT} = −100μA	2.4			V
Logic "0" Output Voltage	I_{OUT} = 1.6mA			0.4	V

ac electrical characteristics [T_A = 25°C, V_{CC} = +5V ± 5%, TTL loading = 100pF (Note 4)]

Parameter	Conditions	Min.	Typ. (Note 2)	Max.	Units
f_X		0.1		4.0	MHz
T_C (Note 5)		500			ns
Microcycle		1			μs
External Clock Input (see figure 2A) T_{W0}		75			ns
T_{W1}			160		ns
XOUT/ADS Timing Relationship (see figure 3) T_H (ADS)			50		ns
Address and Input/Output Status (see figures 5 and 6) T_{D1} (ADS)			$3T_C/2$		ns
T_W (ADS)			$(T_C/2) - 50$		ns
T_S (ADDR)			$(T_C/2) -150$		ns
T_H (ADDR)			50		ns
T_S (STAT)			$(T_C/2) - 100$		ns
T_H (STAT)			50		ns
T_H (NBREQ)		0			ns
Data Input Cycle (see figure 5) T_D (RDS)		0			ns
T_W (RDS)			$T_C + T_{WO} - 75$		ns
T_S (RD)			250		ns
T_H (RD)			0		ns
T_{ACC} (RD)			$2T_C - 300$		ns
Data Output Cycle (see figure 6) T_D (WDS)			$T_C - 50$		ns
T_W (WDS)			$T_C - 50$		ns
T_S (WD)			$(T_C/2) - 150$		ns
T_H (WD)			100		ns
Input/Output Cycle Extend (see figure 7) T_S (HOLD)			300		ns
T_{D1} (HOLD)			200		ns
T_{D2} (HOLD)			200		ns
T_W (HOLD)				∞	ns
Bus Access (see figure 4) T_D (NENOUT)			300		ns
T_{D2} (ADS)			T_C		ns
Output Load Capacitance XOUT				30	pF
All Other Pins (except V_{CC} and GND)				75	pF

Note 1: Maximum ratings indicate limits beyond which damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under electrical characteristics.

Note 2: Typical parameters correspond to nominal supply voltage at 25°C.

Note 3: NBREQ is an input/output signal that requires an external resistor to V_{CC}.

Note 4: All times measured from valid Logic "0" level = 0.8V or valid Logic "1" level = 2.0V.

Note 5: T_C is the time period for two clock cycles of the on-chip or external oscillator ($T_C = 2/f_X$). Refer to paragraph titled Timing Control for detailed definition.

functional description

SC/MP is a self-contained general-purpose microprocessor designed for ease of implementation in stand-alone, DMA (Direct Memory Access), and multiprocessor applications. Communications between SC/MP and external memory/peripheral devices are effected via a 12-bit dedicated address bus and an 8-bit bidirectional data bus. During the address interval of each input/output cycle, SC/MP employs both busses to provide a 16-bit address output: the 12 least significant address bits are sent out over the 12-bit address bus and the 4 most significant address bits are sent out over the 8-bit data bus along with 4 status bits. Separate strobe outputs from SC/MP (NADS, NWDS, NRDS) indicate when valid address information is present on the two busses, and when valid input/output memory or peripheral data are present on the 8-bit bus. To further extend flexibility of application, serial data input/output ports are also provided so that serial data transfers can be effected under program control. The remaining input/output signals shown in figure 1 are dedicated to general-purpose control and status functions, including initialization, bus management, microprocessor halt, interrupt request, input/output cycle extension, and user-specified hardware/software interface functions. A detailed description of each input/output signal is provided in table 1.

FIGURE 1. SC/MP-II Detailed Block Diagram

TABLE 1. Input/Output Signal Description

Signal Mnemonic	Functional Name	Description
NRST	Reset Input	Set high for normal operation. When set low, aborts in-process operations. When returned high, internal control circuit zeroes all programmer-accessible registers; then, first instruction is fetched from memory location 0001_{16}.
CONT	Continue Input	When set high, enables normal execution of program stored in external memory. When set low, SC/MP operation is suspended (after completion of current instruction) without loss of internal status.
NBREQ	Bus Request Input/Output	Associated with SC/MP internal allocation logic for system bus. Can be used as bus request output or bus busy input. Requires external load resistor to V_{CC}.
NENIN	Enable Input	Associated with SC/MP internal allocation logic for system bus. When set low, SC/MP is granted access to system busses. When set high, places system busses in high-impedance (TRI-STATE®) mode.
NENOUT	Enable Output	Associated with SC/MP internal allocation logic for system bus. Set low when NENIN is low and SC/MP is not using system busses (NBREQ-high). Set high at all other times.
NADS	Address Strobe Output	Active-low strobe. While low, indicates that valid address and status output are present on system busses.
NRDS	Read Strobe Output	Active-low strobe. On trailing edge, data are input to SC/MP from 8-bit bidirectional data bus. High-impedance (TRI-STATE®) output when input/output cycle is not in progress.
NWDS	Write Strobe Output	Active-low strobe. While low, indicates that valid output data are present on 8-bit bidirectional data bus. High-impedance (TRI-STATE®) output when input/output cycle not in progress.
NHOLD	Input/Output Cycle Extend Input	When set low prior to trailing edge of NRDS or NWDS strobe, stretches strobe to extend input/output cycle; that is, strobe is held low until NHOLD signal is returned high.
SENSE A	Sense/Interrupt Request Input	Serves as interrupt request input when SC/MP internal IE (Interrupt Enable) flag is set. When IE flag is reset, serves as user-designated sense condition input. Sense condition testing is effected by copying status register to accumulator.
SENSE B	Sense Input	User-designated sense-condition input. Sense-condition testing is effected by copying status register to accumulator.
SIN	Serial Input to E Register	Under software control, data on this line are right-shifted into E register by execution of SIO instruction.
SOUT	Serial Output from E Register	Under software control, data are right-shifted onto this line from E register by execution of SIO instruction. Each data bit remains latched until execution of next SIO instruction.
FLAGS 0, 1, 2	Flag Outputs	User-designated general-purpose flag outputs of status register. Under program control, flags can be set and reset by copying accumulator to status register.
AD00-AD11	Address Bit 00 through Address Bit 11	Twelve TRI-STATE® address output lines. SC/MP outputs 12 least significant address bits on this bus when NADS strobe is low. Address bits are then held valid until trailing edge of read (NRDS) or write (NWDS) strobes. After trailing edge of NRDS or NWDS strobe, bus is set to high-impedance (TRI-STATE®) mode until next NADS strobe.

NOTE:

The 8-bit bidirectional data bus is set to the high-impedance (TRI-STATE®) mode except when it is actually in use by SC/MP (NADS, NRDS, or NWDS low). During the addressing interval of each input/output cycle (NADS low), SC/MP provides address and status outputs over the bus; during the ensuing data-transfer interval (NRDS or NWDS low), 8-bit input or output data bytes are routed over the bus.

TABLE 1. Input/Output Signal Description (Continued)

Signal Mnemonic/ Pin Designation	Output at NADS Time		Input at NRDS Time	Output at NWDS Time
	Functional Name	Description		
DB0	Address Bit 12	Fourth most significant bit of 16-bit address.	Input data are expected on the eight (DB0-DB7) lines.	Output data are valid on the eight (DB0-DB7) lines.
DB1	Address Bit 13	Third most significant bit of 16-bit address.		
DB2	Address Bit 14	Second most significant bit of 16-bit address.		
DB3	Address Bit 15	Most significant bit of 16-bit address.		
DB4	R-Flag	When high, data input cycle is starting; when low, data output cycle is starting.		
DB5	I-Flag	When high, first byte of instruction is being fetched.		
DB6	D-Flag	When high, indicates delay cycle is starting; that is, second byte of DLY instruction is being fetched.		
DB7	H-Flag	When high, indicates that Halt Instruction has been executed. (In some system configurations, the H-Flag output is latched and, in conjunction with the CONTinue input, provides a programmed halt.)	**Note:** The DB0 through DB7 (AD12-HFLG) lines are a high-impedance (open circuit) load when SC/MP does not have access to the input/ output bus.	

DRIVERS AND RECEIVERS

Equivalent circuits for SC/MP drivers and receivers are shown below. All inputs have static charge protection circuits consisting of an RC filter and voltage clamp. These devices still should be handled with care, as the protection circuits can be destroyed by excessive static charge.

SC/MP-II Driver and Receiver Equivalent Circuits

TIMING CONTROL

All necessary timing signals are provided by a three-stage inverter ring oscillator contained on the SC/MP chip. Two control pins, XIN and XOUT, permit the frequency of the oscillator to be controlled by any of the following methods:

1. By leaving the XOUT pin unterminated and driving the XIN pin with an externally generated TTL clock that conforms to the parameters shown in figure 2A. For this method, the frequency of the oscillator is equal to the frequency of the external clock input.

2. By connecting a resistor-capacitor feedback network between the XIN and XOUT pins and GND as shown in figure 2B.

3. By connecting a crystal with low-pass filter network between the XIN and XOUT pins and GND as shown in figure 2C (for above 1 megahertz) or figure 2D (for 1 megahertz or below). For this method, the frequency of the oscillator is equal to the resonant frequency of the crystal and the low-pass filter prevents unwanted harmonic oscillations.

In addition to illustrating appropriate frequency-control networks for the on-chip oscillator, figures 2A through 2D also show how an optional driver may be used to derive a system clock from the oscillator signal present at the XOUT pin. For reference purposes, the timing relationship between the XOUT signal and the NADS strobe is shown in figure 3.

In the discussions that follow, instruction execution and input/output timing are described in terms of microcycles.

The time interval of a microcycle is four times the period of the oscillator; that is:

period of one microcycle = $2T_C$

$$T_C = 2\left(\frac{1}{f_{osc}}\right) = 2\left(\frac{1}{f_{res}}\right) = 2\left(\frac{1}{f_{XIN}}\right)$$

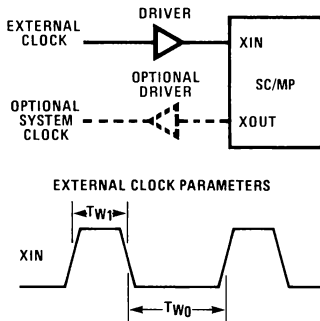
where:

T_C = time period for two cycles of on-chip or external oscillator

f_{osc} = frequency of on-chip oscillator

f_{res} = resonant frequency of crystal connected between XIN and XOUT pins

f_{XIN} = frequency of external clock applied to XIN pin

A. External Clock Input

B. Resistor-Capacitor Feedback Network

C. Crystal with Low-Pass Filter (Above 1 MHz)

Suggested values for Crystal with Low-Pass Filter Network:

Crystal	Rp	C_1	R_1
2MHz	1MΩ	20pF	3kΩ
3.58MHz	100kΩ	20pF	1kΩ
4MHz	100kΩ	200pF	1kΩ

D. Crystal with Low-Pass Filter (1 MHz or Below)

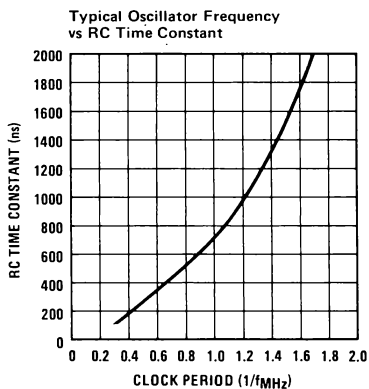
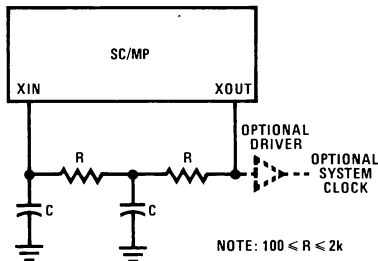



FIGURE 2. Frequency Control Networks for On-Chip Oscillator

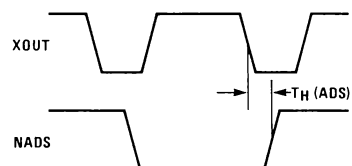
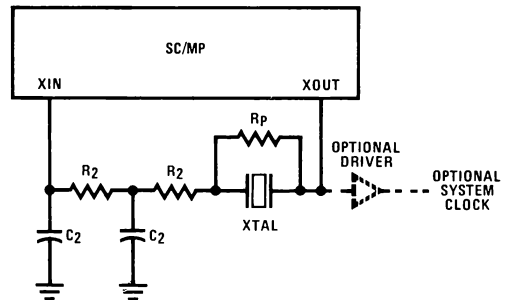



FIGURE 3. XOUT/NADS Timing Relationship

INSTRUCTION FORMAT

The SC/MP instruction repertoire includes both single-byte and double-byte instructions. A single-byte instruction consists of an 8-bit operation code that specifies an operation that SC/MP can execute without further reference to memory. A double-byte instruction consists of an 8-bit operation code and an 8-bit data or displacement field. When the second byte represents a data field, the data are processed by SC/MP during execution of the instruction, thereby eliminating the need for further memory references. When the second byte represents a displacement value, it is used to calculate a memory address that will be accessed (written into or read from) during execution of the instruction (refer to Addressing).

DATA STORAGE

As shown in figure 1, SC/MP provides ten internal registers, seven of which are accessible to the programmer. The purpose and function of these registers are described below.

Program Counter – The program counter is a 16-bit register that contains the address of the instruction being executed. The contents of this register are automatically incremented by one just before each instruction is fetched from memory to enable sequential execution of the stored instructions. Under program control, the contents of this register also may be modified or exchanged with the contents of a pointer register to effect subroutine calls and program branches.

NOTE:

The 16-bit address output of the program counter consists of a 4-bit high-order address and a 12-bit low-order address. When the program counter is incremented at the start of each instruction fetch input/output cycle, only the 12 low-order bits are affected; no carry is provided to the 4 high-order bits. For systems employing memories of 4k or less, the high-order bits can be ignored as they are set to 0000_{16} following initialization. For systems employing larger memories, the contents of a pointer register can be modified to select the desired 4k block of memory.

Pointer Registers – The pointer registers are 16-bit general-purpose registers that normally are loaded under program control with reference addresses that serve as page pointers, stack pointers, and subroutine pointers. In applications having minimal memory addressing requirements, these registers may be used alternately as data storage registers.

NOTE:

When interrupt requests are enabled, pointer register 3 is automatically referenced by the internal microprogram for formation of the starting address of the user-generated interrupt service routine. (See figure 9.) In this case, the contents of pointer register 3 must be set to one less than the memory location of the first instruction in the interrupt service routine.

Accumulator – The 8-bit accumulator (AC) is the primary working register of SC/MP. It is used for performing and storing the results of arithmetic and logic operations as well as for data transfers, shifts, rotates, and data exchanges with the program counter, the pointer registers, and the status register.

Extension Register – The extension register is used both for serial input/output data transfers and with the accumulator to effect arithmetic, logic, and data-transfer operations. If the second byte of an indexed or auto-indexed memory-reference instruction (refer to Addressing) equals -128_{10}, the contents of the extension register are used as the displacement value for address formation.

Status Register – The status register provides storage for arithmetic, control, and software status flags. For more-detailed information on the function of this register, refer to Status Register under the description of the Arithmetic and Logic Unit.

Instruction Register – The 8-bit instruction register is not accessible to the programmer. During the fetch phase of each instruction cycle, this register is loaded with the 8-bit instruction operation code retrieved from memory (for a single-byte instruction or the first byte of a double-byte instruction).

Data Input/Output Register – The data input/output register is not accessible to the programmer. It is used for temporary storage of all input/output data received via or transmitted over the 8-bit bidirectional data bus during the data-transfer interval of each input/output cycle (NRDS or NWDS low).

Address Register – The 16-bit address register is not accessible to the programmer. It is used for temporary storage of the 16-bit address transmitted during an input/output cycle.

ARITHMETIC AND LOGIC UNIT

The Arithmetic and Logic Unit (ALU) provides the data-manipulation capability that is an essential feature of any microprocessor. The operations provided by the ALU include OR, XOR, increment, decrement, binary addition, and decimal addition. For decimal addition, the data inputs to the ALU are treated as two 4-bit BCD digits, thereby eliminating the program-storage and execution time required to perform BCD to binary conversion.

BUS TRANSFER LOGIC

The bus transfer logic processes the gating and function control outputs of the instruction-decode logic to provide the shift-right (with link, without link, or with serial input data), rotate (with or without link), and bus-exchange functions necessary for data movement between the SC/MP internal read and write busses. A general summary of the data-manipulation capabilities available to the programmer follows.

1. Either the low-order or the high-order byte of any pointer register can be exchanged with the contents of the 8-bit accumulator. Thus, data exchanges between the pointer registers can be effected one byte at a time via the accumulator.

2. The contents of the program counter can be directly exchanged with the contents of any pointer register.

3. The contents of the extension register can be loaded into the accumulator or can be exchanged with the contents of the accumulator. When the accumulator is loaded from the extension register, the original contents of the accumulator are lost.

4. The contents of the status register can be copied into the accumulator to enable status modification or conditional-branch testing. When the status register is copied into the accumulator, the contents of the status register are not altered but the original contents of the accumulator are lost.

5. The contents of the accumulator can be copied into the status register to change the outputs of the status register, except for status bits 4 and 5 (Sense A and B inputs to SC/MP). Since these are read-only bits, they are not affected by data movements internal to SC/MP. Copying the accumulator into the status register does not alter the contents of the accumulator.

NOTE:

The flag 0, 1, and 2 outputs of the status register serve as latched flags; in other words, they are set to the specified state when the contents of the accumulator are copied into the status register, and they remain in the specified state until the contents of the status register are modified again under program control.

STATUS REGISTER

The function of each bit in the status register is described briefly below.

7	6	5	4	3	2	1	0
CY/L	OV	S_B	S_A	IE	F_2	F_1	F_0

User Flag 0 – User-assigned general-purpose status bit for implementation as software status bit or in system control applications. This status bit is available as an external output from SC/MP.

User Flag 1 – Same as User Flag 0.

User Flag 2 – Same as User Flag 0.

Interrupt Enable Flag – Internal status bit that is set and reset under program control. When set, SC/MP recognizes external interrupt requests received via Sense A input. When reset, inhibits SC/MP from recognizing interrupt requests.

Sense A – General-purpose status input for sensing external conditions. When IE flag is reset, this bit can be tested by copying status register to accumulator. When IE flag is set, this bit serves as interrupt request input causing SC/MP to automatically branch to user-generated interrupt-service routine in response to high input.

Sense B – Same as Sense A except that it is not tested for interrupt status.

NOTE:

Sense A and B inputs are read-only bits. Thus, they are not affected when the contents of the accumulator are copied into the status register.

Overflow (OV) – This bit is set if an arithmetic overflow occurs during an add (ADD, ADI, or ADE) or a complement-and-add instruction (CAD, CAI, or CAE). It is not affected by the decimal-add instructions (DAD, DAI, or DAE).

Carry/Link (CY/L) – This bit is set if a carry from the most significant bit occurs during an add, complement-and-add, or decimal-add instruction. Thus, it serves as a carry input to the next add instruction. In addition, it is included in the Shift Right with Link (SRL) and Rotate Right with Link (RRL) instructions.

CONTROL

The operation of the SC/MP microprocessor consists of repeatedly accessing or fetching instructions from the program stored in external memory and executing the operations specified by the instructions. These two steps are carried out under the control of an internal microprogram. (SC/MP is not user-microprogrammable.) The microprogram is similar to a state table specifying the series of states of system control signals necessary to carry out each instruction. Microprogram storage is provided in the instruction decode and control logic, and microprogram routines are implemented to fetch and execute instructions. The fetch routine first increments the program counter, and then causes the instruction address to be transferred from the program counter to the system busses via the output address register. The microprogram next initiates an input data transfer. When the instruction operation code is subsequently placed on the 8-bit data bus (single-byte instruction or first byte of double-byte instruction), the operation code is loaded into the instruction register. The operation code is then partially decoded to determine whether the instruction contains a second byte. If it does, a second input data transfer is effected to load the next byte in the data input/output register.

After the complete instruction is stored in the instruction and/or data input/output register(s), the instruction decoder transforms the instruction operation code into the address of the appropriate instruction-execution routine contained in the internal microprogram. The microprogram then branches to the specified internal address to initiate execution of the instruction. The resulting execution routine comprises one or more microinstructions that implement the required functions. For example, the first microcycle of an Extension Register Add Instruction (ADE) causes the contents of the extension register to be gated onto the read bus, transferred to the write bus via the bus control logic, and then written into the data input/output register. The next microcycle causes the contents of the accumulator to be gated onto the read bus, the contents of the read bus to be added to the contents of the data input/output register via the ALU, and the resultant output of the ALU to be written into the accumulator via the write bus. The final step of the execution routine is a jump back to the fetch routine to access the next instruction.

INITIALIZATION

Since SC/MP may power up in a random condition, the following power-up and initialization procedure is recommended.

1. Apply power (GND and V_{CC}) and set NRST low.

NOTE:

Allow ample time (typically, 100ms) for the oscillator and the internal clocks to stabilize. In systems where NRST is set low after turning on power, NRST must remain low for a minimum of $4T_C$. While NRST is low, any in-process operations are aborted automatically. When NRST is low, strobes and address and data busses are in the Non-I/O state (high-Z state).

2. Set NRST high.

NOTE:

This causes the SC/MP internal control circuit to set the contents of all programmer-accessible registers to zero. Thus, when SC/MP is granted access to the system busses following initialization, the first instruction is fetched *always* from memory location 0001_{16}. The NBREQ output goes low, indicating the start of this input/output cycle; this occurs at a time within $13T_C$ after NRST is set high. Normal execution of the program continues as long as NRST remains high.

parallel data transfers

Parallel data transfers occur during each instruction fetch and during the ensuing read/write cycle associated with execution of the memory-reference instructions. This class of instruction could perhaps more properly be called the "Input/Output Reference Class" in the case of the SC/MP microprocessor, since all data transfers, whether with memory, peripheral devices, or a central processor data bus, occur through the execution of these instructions. This unified bus structure is in contrast with many other microprocessors and minicomputers that have one instruction type (input/output class) for communication with peripheral devices and another instruction type (memory reference class) for communication with memories. The advantage of the approach taken by SC/MP is that a wider variety of instructions (the entire memory-reference class) is available for communications with peripherals. Thus, the LD and ST (Load and Store) instructions can be used for basic transfers, the ILD and DLD (increment/decrement and load) instructions can be used for indexing peripheral registers, and the remaining memory reference instructions can be used, as required, for "one-step" retrieval and processing of peripheral input data.

BUS ACCESS

Before SC/MP can initiate parallel data transfers with memory or peripheral devices, it must have access to the system address and data busses. Three of the SC/MP input/output signals are associated with bus control: NBREQ, NENIN, and NENOUT. For simple stand-alone applications, the NENOUT signal can be ignored and the NENIN signal can be tied to GND to allow the SC/MP microprocessor to have continual access to the system busses. The NBREQ input/output line then goes low during each input/output cycle as shown in figures 5 and 6 to indicate when SC/MP is actually using the system busses.

NOTE:

The NBREQ input/output line must be tied to V_{CC} via an external load resistor to allow normal operation of the SC/MP microprocessor.

For DMA and multiprocessor applications, the NBREQ, NENIN, and NENOUT signals can be interconnected in various configurations to allow bus access to be granted to requesting devices according to user-specified priorities. Figure 4 illustrates the general sequence in which these signals are processed by SC/MP to gain access to the system busses and to indicate when the busses are actually being used.

INPUT/OUTPUT CYCLE

Once SC/MP has control of the system busses, the actual input/output cycle begins. As shown in figures 5 and 6, the functions of memory addressing, data reading, and data writing are implemented, respectively, by the address strobe (NADS), the read strobe (NRDS), and the write strobe (NWDS). Note that the NBREQ signal is reset high at the end of the input/output cycle to indicate that the system busses are now free for use by the highest-priority requesting device.

The first operation that SC/MP performs for each input/output cycle is to load the 12 least significant address bits onto the 12-bit address bus, and the 4 most significant address bits along with 4 status bits onto the 8-bit data bus. At the same time, SC/MP sets the NADS output low to indicate that the address and the status information are valid. The low-order address on the 12-bit bus is then held valid for the duration of the input/output cycle; the high-order address and the status information on the 8-bit bus remain valid only while NADS is low. While valid, the status bits have the following significance:

RFLG – When high, indicates that input/output cycle is read cycle; when low, indicates that input/output cycle is write cycle.

IFLG – Set high to indicate that instruction operation code (single-byte instruction or first byte of double-byte instruction) will be output from memory following NADS.

DFLG – Set high only when second byte of Delay Instruction is to be read from memory following NADS. Execution of the Delay Instruction then starts at trailing edge of NRDS. Upon completion, SC/MP provides NADS output to initiate next input/output cycle if bus access is granted. Time in microcycles from leading edge of delay flag to leading edge of subsequent NADS output is computed from the following formula:

$$\text{Delay} = [9 + 2(\text{AC}) + 2\ \text{disp} + 2^9\ \text{disp}]\ \text{microcycles}$$

where

(AC) = unsigned contents of accumulator

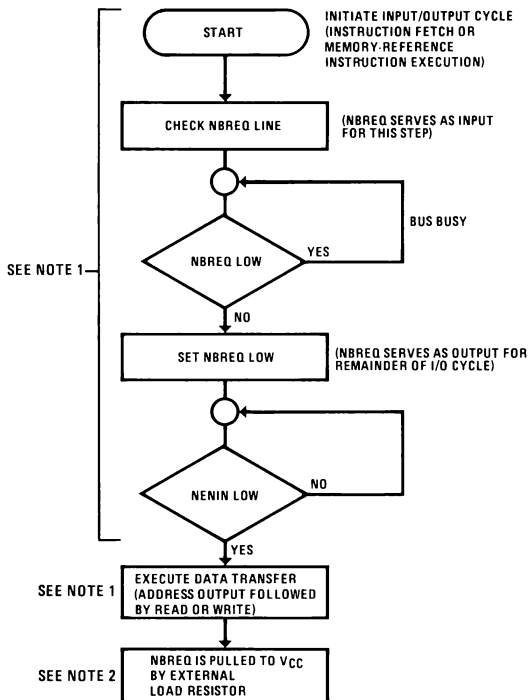
disp = unsigned displacement value contained in second byte of Delay Instruction

The time derived from the above formula does not include the four microcycles required to fetch the first byte of the Delay Instruction. Thus, when the Delay Instruction is used for software timing, total instruction execution time equals $[13 + 2(\text{AC}) + 2\ \text{disp} + 2^9\ \text{disp}]$ microcycles.

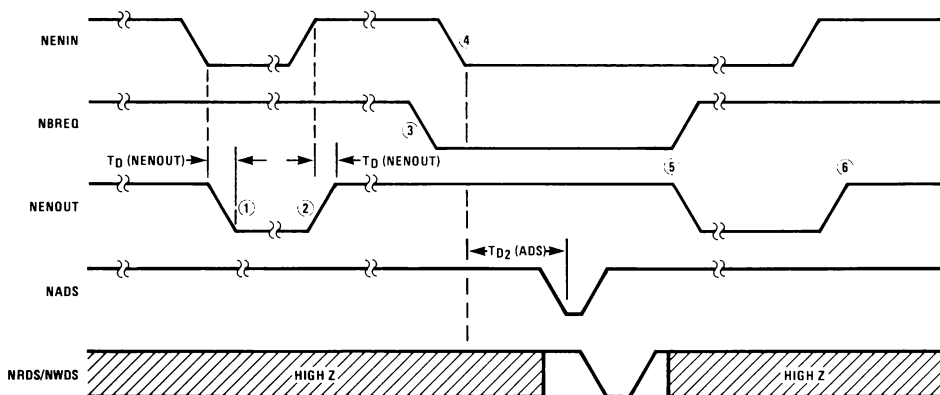
NOTE:

When Halt Instruction is executed, instruction decode and control logic inhibits incrementing of program counter for one input/output cycle. Thus, Halt Instruction is read from memory a second time to enable generation of HFLG output, but no further processing of Halt Instruction occurs. In effect, this procedure ensures HFLG is output in advance of the next instruction to be fetched from memory.

HFLG – Set high only during addressing interval of read cycle that follows Halt Instruction. HFLG may be used to cause user-provided external logic to set the CONT input low, and thereby to effect a programmed halt. Since HFLG read cycle precedes the next instruction

A. NBREQ and NENIN Processing Sequence

Note 1: NENOUT is always high while SC/MP is actually using bus; that is, NENIN input and NBREQ output are low.

Note 2: When SC/MP is not using bus (NBREQ output or NENIN input high), NENOUT is held in same state as NENIN input.

B. NBREQ, NENIN, and NENOUT Timing

Note 1: NENOUT goes low to indicate that SC/MP was granted access to bus (NENIN low) but is not using bus.

Note 2: NENOUT goes high in response to high NENIN input.

Note 3: SC/MP generates bus request; bus access not granted because NENIN high.

Note 4: NENIN goes low. Bus access now granted and input/output cycle actually initiated. If NENIN is set high while SC/MP has access to the bus, the address and data ports will go to the high-impedance (TRI-STATE®) state, but NBREQ will remain low. When NENIN is subsequently set low, the input/output cycle will begin again.

Note 5: Input/output cycle completed. NENOUT goes low to indicate that SC/MP granted access to bus but not using bus. If NENIN had been set high before completion of input/output cycle, NENOUT would have remained high.

Note 6: NENOUT goes high to indicate that system busses are available for use by highest-priority requestor.

FIGURE 4. Bus Access Control

Note: Timing is valid when NENIN is low before NBREQ is set low by SC/MP; see figure 4 for NADS timing when NENIN is set low after NBREQ.

FIGURE 5. SC/MP Data Input Timing

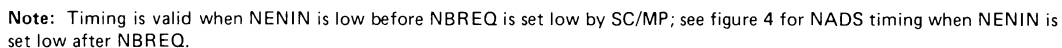



FIGURE 6. Data Output Timing

fetch, termination of programmed halt enables fetch of first instruction that follows Halt Instruction.

After resetting the NADS output, SC/MP generates an NRDS or NWDS strobe, respectively, to initiate a data-input (read) or data-output (write) operation. For a read operation, input data are strobed into SC/MP from the 8-bit bus on the trailing edge of the NRDS strobe. For a write operation, SC/MP places valid output data on the 8-bit bus on the leading edge of the NWDS strobe. After resetting the NRDS or NWDS strobe to complete the data transfer, SC/MP then resets the NBREQ signal to indicate that the system busses are free for use by another controller.

INPUT/OUTPUT CYCLE EXTENSION

As shown in figure 7, the NHOLD signal may be set low prior to the trailing edge of the NRDS or NWDS strobe to cause SC/MP to lengthen the input/output cycle by holding the strobe active until after the NHOLD signal is returned high. Since there is no restriction on the maximum duration of the NHOLD signal, it can be used in a variety of applications ranging from accommodation of memories/peripherals with long access times to single-cycle control of the operating program for software debug purposes.

Figure 8 illustrates a typical circuit that may be used to generate an NHOLD signal of repeatable duration. The circuit shown employs a DM74165 8-Bit Parallel In/Serial Out Shift Register to allow selection of an input/output cycle extend time that ranges from $T_C/2$ to $2T_C$ in increments of $T_C/2$. Functional operation of the circuit is controlled by the NADS strobe and XOUT signals. Each time that the NADS strobe goes low, the data present at the A through H terminals are loaded into the shift register in parallel. When the NADS strobe subsequently returns high, the data are then shifted out serially on the positive-to-negative transitions of XOUT. Thus, the NHOLD output of the circuit is set low on the leading edge of each NADS strobe and, as shown in the chart that accompanies the circuit diagram, it remains low for a time period ranging from three clock cycles minimum (B, C, D, and E inputs = Logic "1") to seven clock cycles maximum (B, C, D, and E inputs = Logic "0").

It is important to note that instruction execution time is increased whenever an input/output cycle is extended via the NHOLD signal. For purposes of computing the increase in instruction execution time, it is necessary to distinguish between the terms *Input/Output Cycle Delay Period* and *Input/Output Cycle Extend Time*. The term *Input/Output Cycle Delay Period* refers to the time that the NRDS/NWDS strobe is actually "stretched" to provide the required memory or peripheral access time. The term *Input/Output Cycle Extend Time* refers to the additional number of microcycles required by the internal SC/MP microprogram to complete the extended input/output cycle; that is:

Input/Output Cycle Delay Period	**Input/Output Cycle Extend Time**
$T_C/2$ through $2T_C$ ($> 0 \leq 1$ μcycle)	1 μcycle
$5T_C/2$ through $4T_C$ ($> 1 \leq 2$ μcycles)	2 μcycles
$9T_C/2$ through $6T_C$ ($> 2 \leq 3$ μcycles)	3 μcycles
etc.	etc.

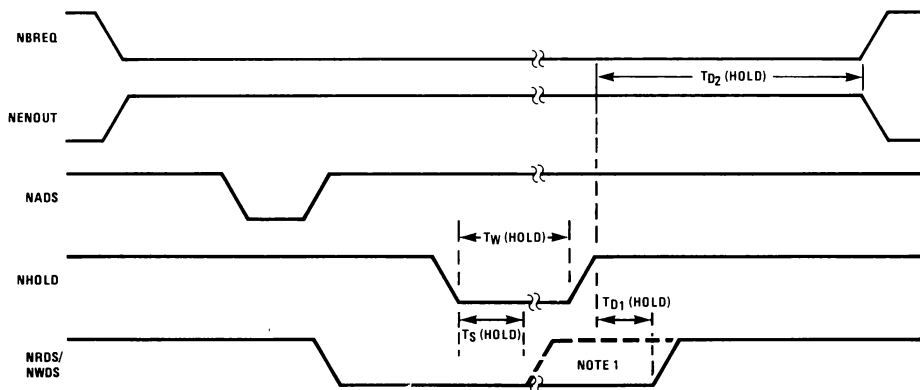
The total increase in instruction execution time, therefore, is equal to the *Input/Output Cycle Extend Time* multiplied by the total number of input/output cycles associated with the instruction. For example, a DLD Instruction is normally executed in 22 microcycles. Since this instruction employs three read input/output cycles and one write input/output cycle, an *Input/Output Cycle Extend Time* of one microcycle would increase total DLD Instruction execution time to 26 microcycles.

Note: Dashed trailing edge of NRDS/NWDS indicates normal strobe timing when NHOLD is not active.

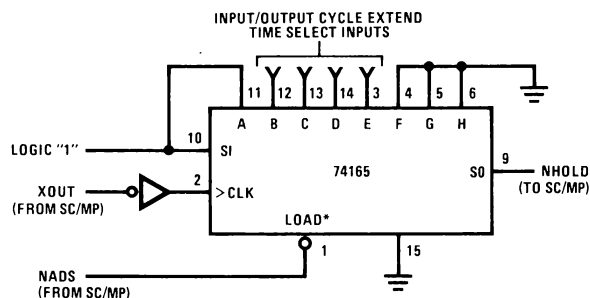
FIGURE 7. Extended Input/Output Timing

Data Inputs E D C B	NHOLD Duration (in clock cycles)	Input/Output Cycle		Required Memory Access Time [T_{ACC} (RD)]
		Delay Period	Extend Time in Microcycles	
1 1 1 1	3	0	0	$2T_C - 300$
1 1 1 0	4	$T_C/2$	1	$(5T_C/2) - 300$
1 1 0 0	5	T_C	1	$3T_C - 300$
1 0 0 0	6	$3T_C/2$	1	$(7T_C/2) - 300$
0 0 0 0	7	$2T_C$	1	$4T_C - 300$

FIGURE 8. Typical NHOLD Control Circuit

serial data transfers

Serial input/output data transfers can be used efficiently with very slow input/output peripherals such as X-Y plotters, teletypewriters, slow-speed printers, and so forth. Such transfers can be effected in any of the following manners:

1. By assigning serial input/output functions to the extension register via the SIO (Serial Input/Output) Instruction. When this instruction is executed, the contents of the extension register are shifted right one bit. At the same time, data present on the SIN line are shifted into bit position 7 of the extension register and the original contents of bit position 0 are shifted into a flip-flop to provide a latched output of the SOUT line. The SOUT data are then held latched until the next SIO instruction is executed.

2. By using one of the status flags as an output data bit and one of the sense lines as an input data bit.

3. By implementing external logic such that only one line of the 8-bit data input/output bus is used.

For synchronous systems, serial data input/output timing may be provided by program loops that employ the delay instruction, or by using one or more of the transfer instructions (see table 2) to test the output of an external timing circuit. For asynchronous systems, one of the sense inputs can be used for testing bit-received/ready status and a pulsed flag output can be provided, under program control, for peripheral indexing each time that a data bit is actually shifted in or out.

Systems that have several input/output devices must be multiplexed; device selection can then be accomplished using the status flag outputs of SC/MP, or by using parallel input/output commands to load an external latch. Systems that do not require serial input/output capability can employ the SIN and SOUT lines as a sense input and flag output, respectively.

interrupts

When the internal interrupt enable (IE) flag is set under program control, the Sense A line is enabled to serve as an interrupt request input; when the IE flag is reset, SC/MP is inhibited from detecting interrupts. Thus, while the IE flag is set, the Sense A input is tested prior to the fetch phase of each instruction as shown in figure 9. Upon detection of an interrupt request (Sense A high), the following events occur automatically.

1. The status register IE flag is reset to prevent SC/MP from responding to any further interrupt requests. Interrupt request capability can then be reenabled during or at the end of the ensuing user-generated interrupt service routine via the IEN (Enable Interrupt) Instruction or by copying the accumulator into the status register.

2. The contents of the program counter are exchanged with the contents of the pointer register 3.

3. The contents of the program counter are incremented by one to address the first instruction of the user-generated interrupt service routine.

The interrupt system must be armed before interrupts are enabled. This is accomplished as follows:

1. First, the Interrupt Enable Bit in the Status Register is set true by executing either an Enable Interrupt Instruction (IEN) or a Copy Accumulator to Status Register Instruction (CAS).

2. Second, one additional instruction is fetched and executed.

A return from interrupt is accomplished by executing two instructions: Enable Interrupt (IEN) immediately followed by Exchange Pointer 3 with Program Counter (XPPC 3).

microprocessor halt

The CONT input to SC/MP is provided to enable suspension of operation without loss of internal status. Processing of the CONT input is shown in figure 9. Since this is an asynchronous input, it can be controlled by external timing logic, or as stated previously, the HALT flag output that appears on the 8-bit data bus (during the read cycle that follows execution of a Halt Instruction) can be used with an external circuit to effect a programmed halt condition. Note that when an interrupt request is detected while the CONT input is low, the first instruction of the user-generated interrupt service routine is automatically executed. Thus, the first instruction of the interrupt service routine can be used to reset the external CONT input logic and, thereby, to terminate the microprocessor halt condition if so desired.

After execution of an instruction, the CONT input must be high for a minimum time of $2T_C$ (1 microcycle) in order to fetch and execute the next instruction.

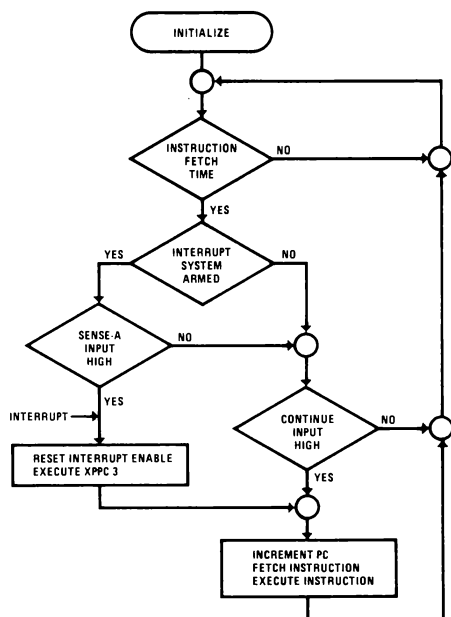



FIGURE 9.

Microprocessor Halt and Interrupt Request Input Processing

instruction set

The SC/MP instruction set provides the general-purpose user of microprocessors a powerful programming capability along with above-average flexibility and speed. The instruction set consists of 46 instructions, which comprise eight general categories. A listing of the complete instruction set is provided in table 2; typical instruction execution times are given in table 3, and notations and symbols used as shorthand expressions of instruction capability are defined in table 4.

ADDRESSING

During execution, instructions and data defined in a program are stored into and loaded from specific memory locations, the accumulator, or selected registers. Because SC/MP, memory (read/write and read-only), and peripherals are on a common data bus, any instruction used to address memory may be used to address the peripherals. The formats of the instruction groups that reference memory are shown below.

7, , , ,3	2	1,0	7, , , , , , ,0	
opcode	m	ptr	disp	Memory Reference Instructions
opcode		ptr	disp	Memory Increment/Decrement Instructions and Transfer Instructions

Memory-reference instructions use the PC-relative, indexed, or auto-indexed methods of addressing memory. The memory-increment/decrement instructions and the transfer instructions use the PC-relative or indexed methods of addressing.

The various methods of addressing memory and peripherals are shown below.

Immediate addressing is an addressing format specific to the immediate instruction group.

Type of Addressing	Operand Formats		
	m	ptr	disp
PC-relative	0	0	-128 to +127
Indexed	0	1, 2, or 3	-128 to +127
Immediate	1	0	-128 to +127
Auto-indexed	1	1, 2, or 3	-128 to +127

For PC-relative, indexed, and auto-indexed memory-reference instructions, another feature of the addressing architecture is that the contents of the extension register are substituted for the displacement if the instruction displacement equals -128 (-X'80).

NOTE:

All arithmetic operations associated with address formation affect only the 12 low-order address bits; no carry is provided to the 4 high-order bits. For systems employing memories of 4k or less, the high-order bits can be ignored as they are set to 0000 following initialization. For systems employing larger memories, the high-order bits must be set to the starting address of the desired 4k block of memory. For example:

0001_2 enables memory locations 1000_{16} - $1FFF_{16}$ to be addressed.

0010_2 enables memory locations 2000_{16} - $2FFF_{16}$ to be addressed and so forth.

PC-Relative Addressing — A PC-relative address is formed by adding the displacement value specified in the operand field of the instruction to the current contents of the program counter. The displacement is an 8-bit twos-

complement number, so the range of the PC-relative addressing format is -128_{10} to $+127_{10}$ locations from the current contents of the program counter.

Immediate Addressing — Immediate addressing uses the value in the second byte of a double-byte instruction as the operand for the operation to be performed (see below).

For example, compare a Load (LD) instruction to a Load Immediate (LDI) instruction. The Load instruction uses the contents of the second byte of the instruction in computing the effective address of the data to be loaded. The Load Immediate instruction uses the contents of the second byte as the data to be loaded.

Indexed Addressing — Indexed addressing enables the programmer to address any location in memory through the use of the pointer register and the displacement. When indexed addressing is specified in an instruction, the contents of the designated pointer register are added to the displacement to form the effective address. The contents of the pointer register are not modified by indexed addressing.

Auto-Indexed Addressing — Auto-indexed addressing provides the same capabilities as indexed addressing along with the ability to increment or decrement the designated pointer register by the value of the displacement. If the displacement is less than zero, the pointer register is decremented by the displacement before the contents of the effective address are fetched or stored. If the displacement is equal to or greater than zero, the pointer register is used as the effective address, and the pointer register is incremented by the displacement after the contents of the effective address are fetched or stored.

system implementation

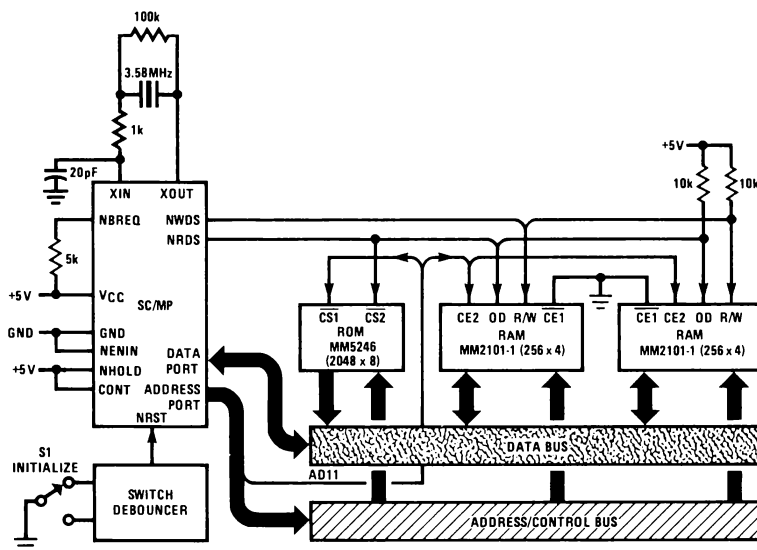
Figures 10 through 12 illustrate typical SC/MP system configurations. In figure 10, SC/MP is shown interconnected to three memory devices to form a stand-alone 4-device system that provides 256 words of read/write memory and 2,048 words for program storage. Figure 11 shows SC/MP interconnected to an external controller for Direct Memory Access (DMA) operation, and figure 12 illustrates a multiprocessor application using SC/MP's built-in logic to control bus access.

NOTE: PART NUMBERS ARE SHOWN ONLY FOR INFORMATION PURPOSES. OTHER MEMORY COMPONENTS WITH SUITABLE CHARACTERISTICS CAN BE USED.

FIGURE 10. SC/MP-II Four-Chip System

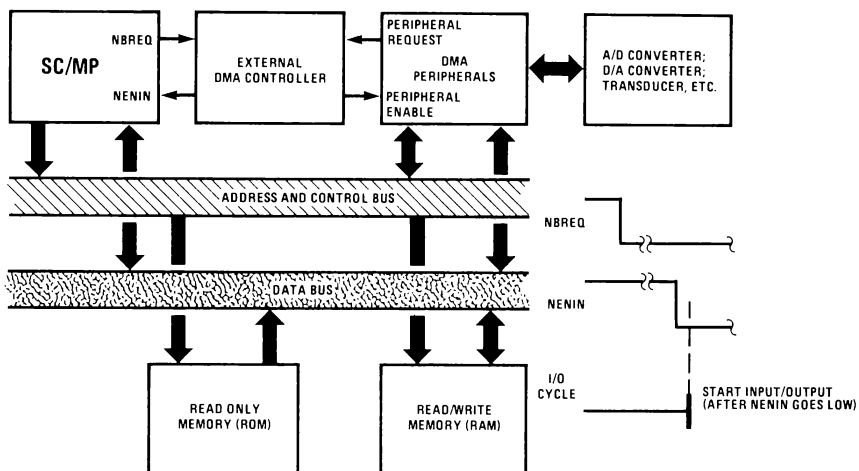



FIGURE 11. SC/MP-II Interconnected for Direct Memory Access (DMA) Operation

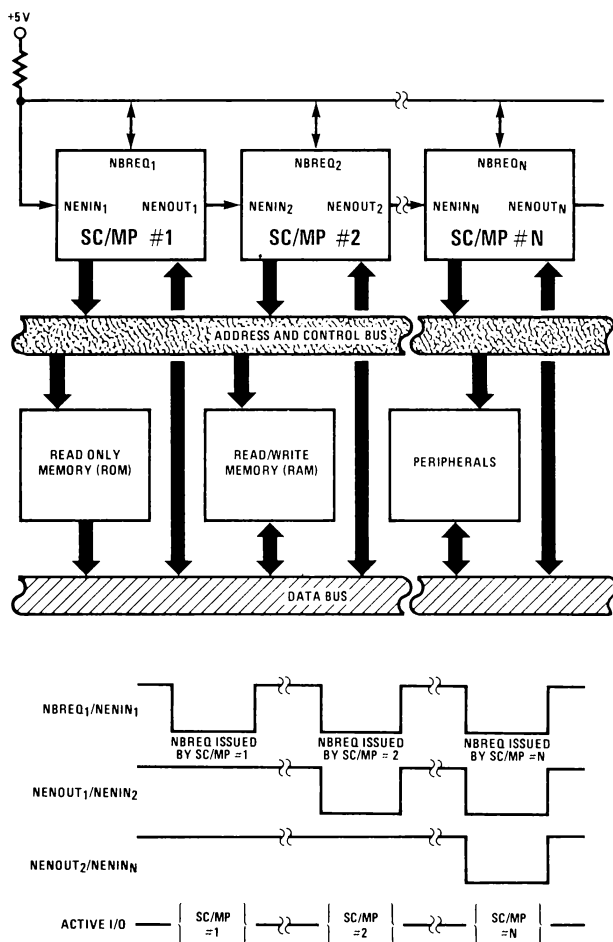



FIGURE 12. Multiprocessor System Using SC/MP-II Built-in Logic for Bus Control

TABLE 2. SC/MP Instruction Summary

DOUBLE-BYTE INSTRUCTIONS

MNEMONIC	DESCRIPTION	OBJECT FORMAT	OPERATION	MICRO-CYCLES
	Memory Reference Instructions	7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0		
LD	Load	1 1 0 0 0 | m | ptr | disp	(AC)←(EA)	18
ST	Store	1 1 0 0 1	(EA)←(AC)	18
AND	AND	1 1 0 1 0	(AC)←(AC) ∧ (EA)	18
OR	OR	1 1 0 1 1	(AC)←(AC) ∨ (EA)	18
XOR	Exclusive-OR	1 1 1 0 0	(AC)←(AC) ⊻ (EA)	18
DAD	Decimal Add	1 1 1 0 1	(AC)←$(AC)_{10}$ + $(EA)_{10}$ + (CY/L);(CY/L)	23
ADD	Add	1 1 1 1 0	(AC)←(AC) + (EA) + (CY/L);(CY/L),(OV)	19
CAD	Complement and Add	1 1 1 1 1	(AC)←(AC) + ~(EA) + (CY/L);(CY/L),(OV)	20
	Memory Increment/Decrement Instructions	7 6 5 4 3 2 | 1 0 | 7 6 5 4 3 2 1 0		
ILD	Increment and Load	1 0 1 0 1 0 | ptr | disp	(AC), (EA)←(EA) + 1	22
DLD	Decrement and Load	1 0 1 1 1 0	(AC), (EA)←(EA) – 1	22
	Immediate Instructions	7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0		
LDI	Load Immediate	1 1 0 0 0 1 0 0 | data	(AC)← data	10
ANI	AND Immediate	1 1 0 1 0 1 0 0	(AC)←(AC) ∧ data	10
ORI	OR Immediate	1 1 0 1 1 1 0 0	(AC)←(AC) ∨ data	10
XRI	Exclusive-OR Immediate	1 1 1 0 0 1 0 0	(AC)←(AC) ⊻ data	10
DAI	Decimal Add Immediate	1 1 1 0 1 1 0 0	(AC)←$(AC)_{10}$ + $data_{10}$ + (CY/L);(CY/L)	15
ADI	Add Immediate	1 1 1 1 0 1 0 0	(AC)←(AC) + data + (CY/L);(CY/L),(OV)	11
CAI	Complement and Add Immediate	1 1 1 1 1 1 0 0	(AC)←(AC) + ~data + (CY/L);(CY/L),(OV)	12
	Transfer Instructions	7 6 5 4 3 2 | 1 0 | 7 6 5 4 3 2 1 0		
JMP	Jump	1 0 0 1 0 0 | ptr | disp	(PC)←EA	11
JP	Jump if Positive	1 0 0 1 0 1	If (AC) ≥ 0, (PC)←EA	9, 11
JZ	Jump if Zero	1 0 0 1 1 0	If (AC) = 0, (PC)←EA	9, 11
JNZ	Jump if Not Zero	1 0 0 1 1 1	If (AC) ≠ 0, (PC)←EA	9, 11
	Double-Byte Miscellaneous Instructions	7 6 5 4 3 2 1 0 | 7 6 5 4 3 2 1 0		
DLY	Delay	1 0 0 0 1 1 1 1 | disp	count AC to –1, delay = 13 + 2(AC) + 2 disp + 2^9 disp microcycles	13 to 131,593

SINGLE-BYTE INSTRUCTIONS

MNEMONIC	DESCRIPTION	OBJECT FORMAT	OPERATION	MICRO-CYCLES
	Extension Register Instructions	7 6 5 4 3 2 1 0		
LDE	Load AC from Extension	0 1 0 0 0 0 0 0	(AC)←(E)	6
XAE	Exchange AC and Extension	0 0 0 0 0 0 0 1	(AC)↔(E)	7
ANE	AND Extension	0 1 0 1 0 0 0 0	(AC)←(AC) ∧ (E)	6
ORE	OR Extension	0 1 0 1 1 0 0 0	(AC)←(AC) ∨ (E)	6
XRE	Exclusive-OR Extension	0 1 1 0 0 0 0 0	(AC)←(AC) ⊻ (E)	6
DAE	Decimal Add Extension	0 1 1 0 1 0 0 0	(AC)←$(AC)_{10}$ + $(E)_{10}$ + (CY/L);(CY/L)	11
ADE	Add Extension	0 1 1 1 0 0 0 0	(AC)←(AC) + (E) + (CY/L);(CY/L),(OV)	7
CAE	Complement and Add Extension	0 1 1 1 1 0 0 0	(AC)←(AC) + ~(E) + (CY/L);(CY/L),(OV)	8
	Pointer Register Move Instructions	7 6 5 4 3 2 1 0		
XPAL	Exchange Pointer Low	0 0 1 1 0 0 | ptr	(AC)↔$(PTR_{7:0})$	8
XPAH	Exchange Pointer High	0 0 1 1 0 1	(AC)↔$(PTR_{15:8})$	8
XPPC	Exchange Pointer with PC	0 0 1 1 1 1	(PC)↔(PTR)	7
	Shift, Rotate, Serial I/O Instructions	7 6 5 4 3 2 1 0		
SIO	Serial Input/Output	0 0 0 1 1 0 0 1	(E_i)→(E_{i-1}), SIN→(E_7), (E_0)→SOUT	5
SR	Shift Right	0 0 0 1 1 1 0 0	(AC_i)→(AC_{i-1}), 0→(AC_7)	5
SRL	Shift Right with Link	0 0 0 1 1 1 0 1	(AC_i)→(AC_{i-1}), (CY/L)→(AC_7)	5
RR	Rotate Right	0 0 0 1 1 1 1 0	(AC_i)→(AC_{i-1}), (AC_0)→(AC_7)	5
RRL	Rotate Right with Link	0 0 0 1 1 1 1 1	(AC_i)→(AC_{i-1}), (AC_0)→(CY/L)→(AC_7)	5
	Single-Byte Miscellaneous Instructions	7 6 5 4 3 2 1 0		
HALT	Halt	0 0 0 0 0 0 0 0	Pulse H-flag	8
CCL	Clear Carry/Link	0 0 0 0 0 0 1 0	(CY/L)←0	5
SCL	Set Carry/Link	0 0 0 0 0 0 1 1	(CY/L)←1	5
DINT	Disable Interrupt	0 0 0 0 0 1 0 0	(IE)←0	6
IEN	Enable Interrupt	0 0 0 0 0 1 0 1	(IE)←1	6
CSA	Copy Status to AC	0 0 0 0 0 1 1 0	(AC)←(SR)	5
CAS	Copy AC to Status	0 0 0 0 0 1 1 1	(SR)←(AC)	6
NOP	No Operation	0 0 0 0 1 0 0 0	None	5

TABLE 3. Instruction Execution Time

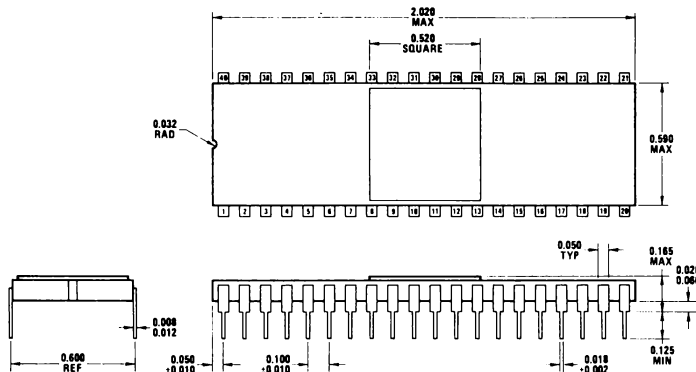
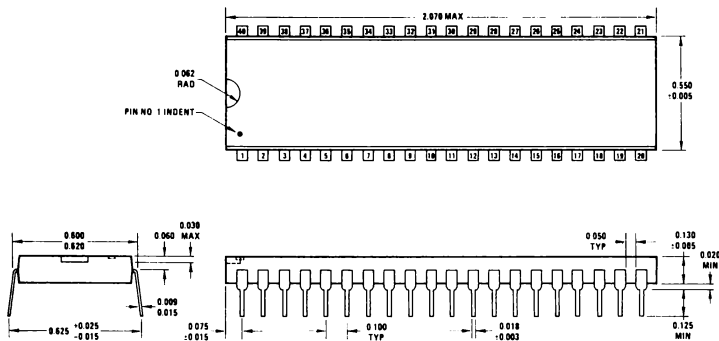
INSTRUCTION	READ CYCLES	WRITE CYCLES	TOTAL MICROCYCLES
ADD	3	0	19
ADE	1	0	7
ADI	2	0	11
AND	3	0	18
ANE	1	0	6
ANI	2	0	10
CAD	3	0	20
CAE	1	0	8
CAI	2	0	12
CAS	1	0	6
CCL	1	0	5
CSA	1	0	5
DAD	3	0	23
DAE	1	0	11
DAI	2	0	15
DINT	1	0	6
DLD	3	1	22
DLY	2	0	13 - 131593
HALT	2	0	8
IEN	1	0	6
ILD	3	1	22
JMP	2	0	11
JNZ	2	0	9, 11 for Jump

INSTRUCTION	READ CYCLES	WRITE CYCLES	MICROCYCLES
JP	2	0	9, 11 for Jump
JZ	2	0	9, 11 for Jump
LD	3	0	18
LDE	1	0	6
LDI	2	0	10
NOP	1	0	5
OR	3	0	18
ORE	1	0	6
ORI	2	0	10
RR	1	0	5
RRL	1	0	5
SCL	1	0	5
SIO	1	0	5
SR	1	0	5
SRL	1	0	5
ST	2	1	18
XAE	1	0	7
XOR	3	0	18
XPAH	1	0	8
XPAL	1	0	8
XPPC	1	0	7
XRE	1	0	6
XRI	2	0	10

Note: If slow memory is being used, the appropriate delay should be added for each read or write cycle.

TABLE 4. Symbols and Notations Used to Express Instruction Execution

SYMBOL AND NOTATION	MEANING
AC	8-bit Accumulator.
CY/L	Carry/Link Flag in the Status Register.
data	Signed, 8-bit immediate data field.
disp	Displacement; represents an operand in a nonmemory reference instruction or an address modifier field in a memory reference instruction. It is a signed twos-complement number.
EA	Effective Address as specified by the instruction.
E	Extension Register; provides for temporary storage, variable displacements and separate serial input/output port.
i	Unspecified bit of a register.
IE	Interrupt Enable Flag.
m	Mode bit, used in memory reference instructions. Blank parameter sets m = 0, @ sets m = 1.
OV	Overflow Flag in the Status Register.
PC	Program Counter (Pointer Register 0); during address formation, PC points to the last byte of the instruction being executed.
ptr	Pointer Register (ptr = 0 through 3). The register specified in byte 1 of the instruction.
$ptr_{n:m}$	Pointer register bits; n:m = 7 through 0 or 15 through 8.
SIN	Serial Input pin.
SOUT	Serial Output pin.
SR	8-bit Status Register.
( )	Means "contents of." For example, (EA) is contents of Effective Address.
[ ]	Means optional field in the assembler instruction format.
~	Ones complement of value to right of ~.
→	Means "replaces."
←	Means "is replaced by."
↔	Means "exchange."
@	When used in the operand field of the instruction, sets the mode bit (m) to 1 for auto-incrementing/auto-decrementing indexing.
10^+	Modulo 10 addition.
$\wedge$	AND operation.
$\vee$	Inclusive-OR operation.
$\forall$	Exclusive-OR operation.
$\geqslant$	Greater than or equal to.
=	Equals.
$\neq$	Does not equal.

physical dimensions

40-Lead Ceramic Dual-in-Line Package (D)

40-Lead Plastic Dual-in-Line Package (N)

ordering information

The SC/MP device may be ordered through the local National Semiconductor sales representative or by contacting our world or international headquarters listed below. The order numbers are as follows:

For an "N" package — ISP-8A/600N
For a "D" package — ISP-8A/600D

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 737-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632

PARTE II

Applicazione del microprocessore SC/MP

Capitolo 1°

In questa seconda parte del volume vengono presentate delle tipiche applicazioni del microprocessore SC/MP.

Ogni applicazione è descritta prima in forma generale, con riferimento all'operatività che si vuole raggiungere, successivamente viene presentato in modo dettagliato il progetto sia hardware che software, corredato di schemi, flowcharts e listing dei programmi.

CONVERSIONE ANALOGICO/DIGITALE E DIGITALE/ANALOGICA

La fig. 1-1 rappresenta il concetto generale di come un sistema basato sul microprocessore SC/MP può utilizzare un convertitore analogico-digitale ed eventualmente un convertitore digitale-analogico. Il generatore del segnale analogico può essere un qualsiasi dispositivo in grado di produrre una corrente od una tensione entro livelli prestabiliti. Sotto controllo del programma il convertitore analogico-digitale campiona il segnale ed il risultato è un numero binario espresso su otto bit. Questo dato viene memorizzato in RAM, dove, a seconda del programma di gestione, può essere utilizzato per vari scopi. Ad esempio, il dato può essere confrontato con un valore di riferimento precedentemente posto in memoria ed a seconda del risultato dell'operazione può essere presa una decisione e realizzata una data operazione di controllo di tipo on/off od eseguita semplicemente una segnalazione per l'operatore. Come altro esempio, la differenza tra il dato acquisito ed il valore di riferimento può essere trattata come segnale di errore e quindi convertito in un segnale analogico tramite un convertitore digitale-analogico. Questo segnale, che è l'equivalente analogico dell'errore calcolato da SC/MP, può essere poi utilizzato dal sistema esterno a scopo di controllo. Inoltre, i dati acquisiti possono essere rappresentati sotto forma alfanumerica ad uso di un operatore.

1-1 Convertitore Analogico/Digitale a singolo ingresso

Descrizione generale

Il convertitore analogico-digitale mostrato in fig. 1-2 richiede pochi componenti, poca memoria ed un programma molto semplice ed è quindi adatto ad applicazioni a basso costo. Nella descrizione che segue sarà data per scontata la conoscenza delle caratteristiche di SC/MP e dei componenti utilizzati, mentre saranno presentate in dettaglio la logica di funzionamento e le operazioni di controllo svolte dal programma.

Operazioni del sistema

Il sistema di conversione analogico-digitale preso in considerazione richiede un impulso di start, un segnale di clock ed un'abilitazione per la presentazione dei dati in uscita.

Supponendo di fornire al convertitore l'opportuno impulso di start della conversione (strt conv), la conversione ha inizio con la prima transizione da "1" a "0" della linea di clock ed ha la durata di 40 impulsi di clock. quando la conversione ha termine, il risultato viene memorizzato in un latch di output e contemporaneamente viene generato il segnale di fine conversione (EOC). Il latch di output è di tipo tri-state e quindi è possibile collegare il convertitore direttamente al bus dei dati di SC/MP (da DB0 a DB7). Quando il processore genera un indirizzo la cui configurazione di bit è valida, viene generato un segnale di output enable per il convertitore (in coincidenza dell'impulso di NRDS). Il risultato della conversione viene quindi letto dal processore, che lo pone in accumulatore.

La fig. 1-3 è il diagramma dei tempi per un ciclo completo di conversione. Come mostrato, la conversione ha inizio con il fronte di discesa del primo impulso di clock, quindi l'impulso di start della conversione deve avere una durata eguale o (meglio) maggiore di quella di un impulso di clock. La fig. 1-3 mostra come, nel

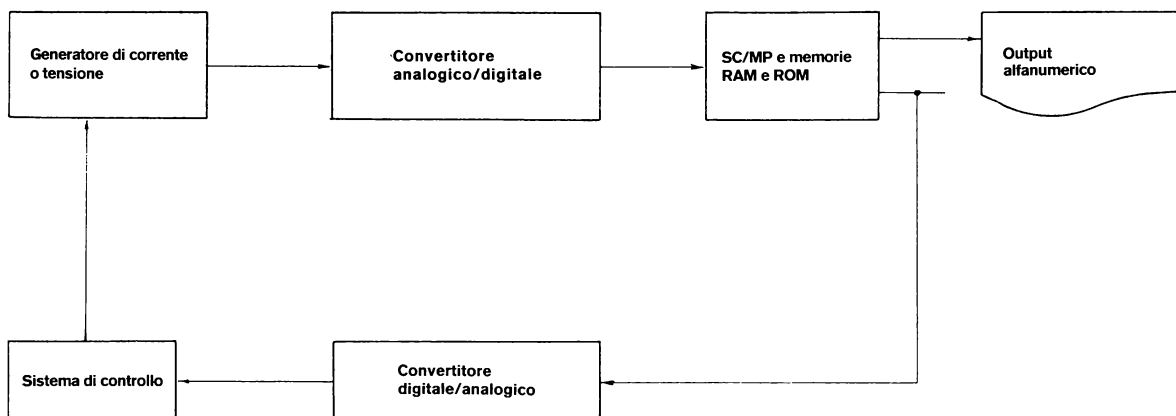



Fig. 1 - 1. Schema a blocchi di un sistema utilizzante conversioni Analogico/Digitale e Digitale/Analogico.

Fig. 1 - 2. Convertitore Analogico/Digitale a singolo ingresso

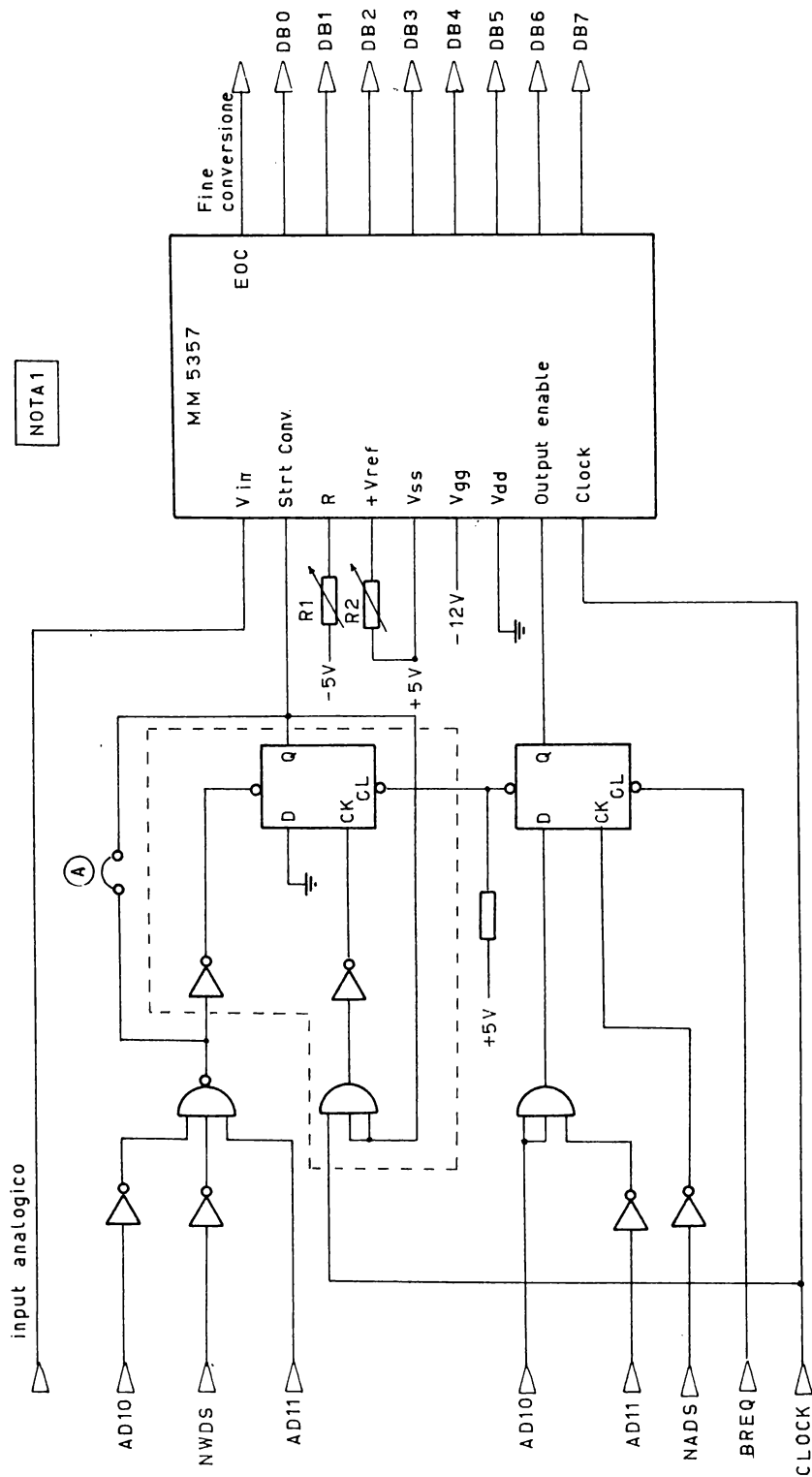
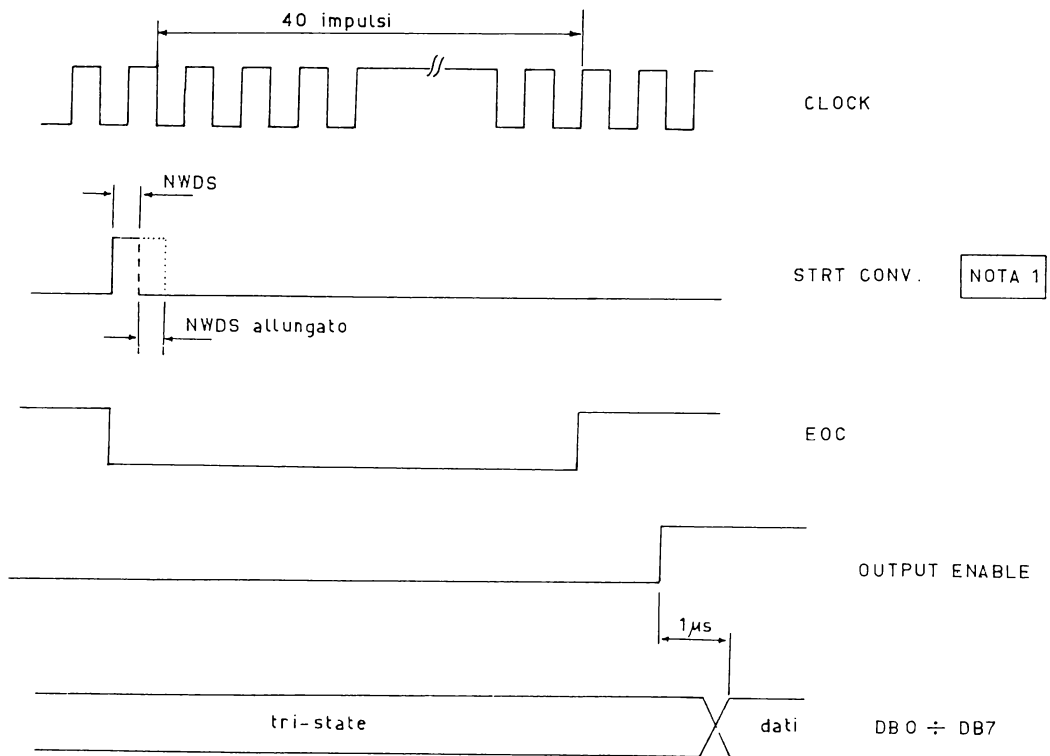



Fig. 1 - 3. Timing del convertitore Analogico/Digitale a singolo ingresso

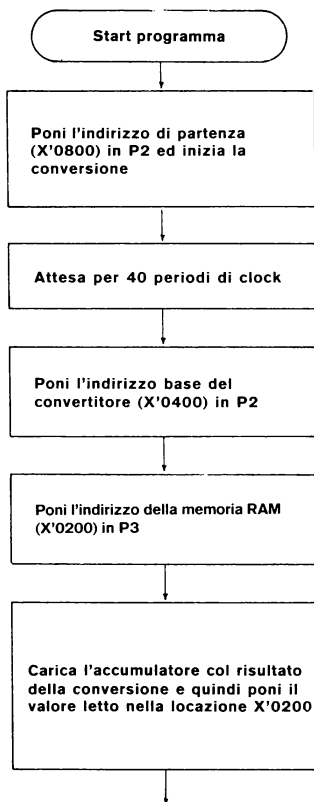
nostro caso la durata dell'impulso di strt conv sia determinata dall'impulso di NWDS di SC/MP. Se il clock fornito al convertitore è più "lento" di NWDS, diventa necessaria la logica descritta in fig. 1-2. Inoltre, in fig. 1-2 è stato scelto come indirizzo di start X'0800, ma si può utilizzare a tale scopo qualsiasi altro indirizzo che non crei conflitti con le altre periferiche o la memoria esistente nel sistema.
Quando ha inizio la conversione, viene testato il segnale analogico presente ed il segnale di EOC (fine della conversione) diventa basso. Durante i successivi 40 impulsi di clock, il segnale viene testato e viene convertito in un valore binario di 8 bit mediante il metodo delle approssimazioni successive. Al termine del quarantesimo impulso di clock, la conversione ha termine, il segnale EOC diventa alto ed il risultato della conversione viene memorizzato nel latch di output (interno al chip di conversione). Quando viene abilitato (segnale di output enable alto) il convertitore presenta in uscita il contenuto del latch di output. Il segnale di output enable viene generato, in questo caso, quando il processore esegue un'istruzione di load (LD) all'indirizzo X'0400. Come si può notare, l'impulso di lettura del convertitore viene generato in coincidenza dell'impulso di NRDS e termina quando la linea di BREQ torna bassa (livello logico "0").

Regolazioni da effettuare

Con le alimentazioni mostrate in fig. 1-2, il convertitore è adatto ad operare con un range di tensione in ingresso di ±5 volts. Le due resistenze R1 e R2 sono comunque necessarie per ottimizzare la precisione del convertitore. R1 provvede all'aggiustamento dello zero e, nel nostro caso, deve essere regolata in modo tale che la transizione da 11111111 a 11111110 avvenga quando il convertitore analogico-digitale ha in ingresso 19,53 millivolt (cioè ad 1/2 del valore del bit meno significativo). R2 invece regola il valore di fondo scala e deve essere posta ad un valore tale per cui la transizione da 00000001 a 00000000 avvenga a 58,6 millivolt dalla tensione di fondo scala (cioè a 9,94 volts circa).

Considerazioni software

Il flowchart ed il listing di fig. 1-4 mostrano come il sistema di conversione analogico-digitale descritto può essere controllato da software per realizzare una funzionalità completa. Allo start del programma, P2 viene caricato col valore X'0800, quindi viene eseguita un'istruzione di load che genera l'impulso di start della conversione. A questo punto, il programma deve la-

sciare tempo al convertitore di ricevere 40 impulsi di clock prima di leggerne il contenuto. In questo caso, sono state aggiunte semplicemente quattro istruzioni di No operation, se però il clock del convertitore dovesse avere una frequenza inferiore ad 1 megahertz, si dovrà introdurre un'istruzione di delay. Una volta iniziata la conversione, P2 viene caricato con l'indirizzo di lettura del convertitore (X'0400) ed il puntatore P3 con l'indirizzo di destinazione RAM del dato che verrà letto (X'0200), quindi, il risultato della conversione viene effettivamente letto e poi posto in memoria.

Note alla fig. 1-2

1 - Se NWDS diventa alto (inattivo) prima che il clock abbia generato un fronte di salita, è necessario realizzare un circuito di allungamento dell'impulso di STRT CONV eguale od equivalente a quello disegnato nel riquadro tratteggiato; in questo caso non si deve eseguire il collegamento A. Se invece l'impulso di NWDS ha una durata sufficiente, si può omettere tutto il circuito nel riquadro e si deve eseguire il collegamento A.

Note alla fig. 1-3

1 - La durata di questo impulso è data dallo strobe di NWDS di SC/MP. Se questo impulso termina nell'istante "a" il sistema non è in grado di operare correttamente. Si deve quindi allungare questo impulso, come mostrato dalla linea tratteggiata (NWDS allungato).

Fig. 1-4 (a)

```
 1          TITLE SCMP, SINGOLO CONVERTITORE A/D
 2
 3
 4
 5  0000  08                 NOP
 6  0001  C 400   START:     LDI    X'00        ;LOAD P2 CON IND. CONVERT.
 7  0003  32                 XPAL   P2
 8  0004  C408               LDI    X'08
 9  0006  36                 XPAH   P2
10  0007  CA00               ST     CONV (P2)   ; START CONVERT.
11  0009  08                 NOP
12  000A  08                 NOP
13  000B  08                 NOP
14  000C  08                 NOP
15  000D  C404    ACCEPT:    LDI    X'04        ; LOAD P2 CON IND. BASE
16                                              ; CONVERT.
17  000F  36                 XPAH   P2
18  0010  C402               LDI    X'02        ; LOAD P3 CON IND. MEMORIA PER
19  0012  37                 XPAH   P3          ; OUTPUT VALORE
20  0013  C400               LDI    X'00
21  0015  33                 XPAL   P3
22  0016  C200               LD     CONV (P2)   ; READ RISULTATO CONVERSIONE
23  0018  C800    STORE:     ST     (P3)        ; SCRITTURA VALORE NELLA
24                                              ; LOCAZIONE DI MEMORIA 200
```

Fig. 1-4 (b) Flow Chart (a) e listing (b) del programma di gestione di un singolo convertitore Analogico/Digitale

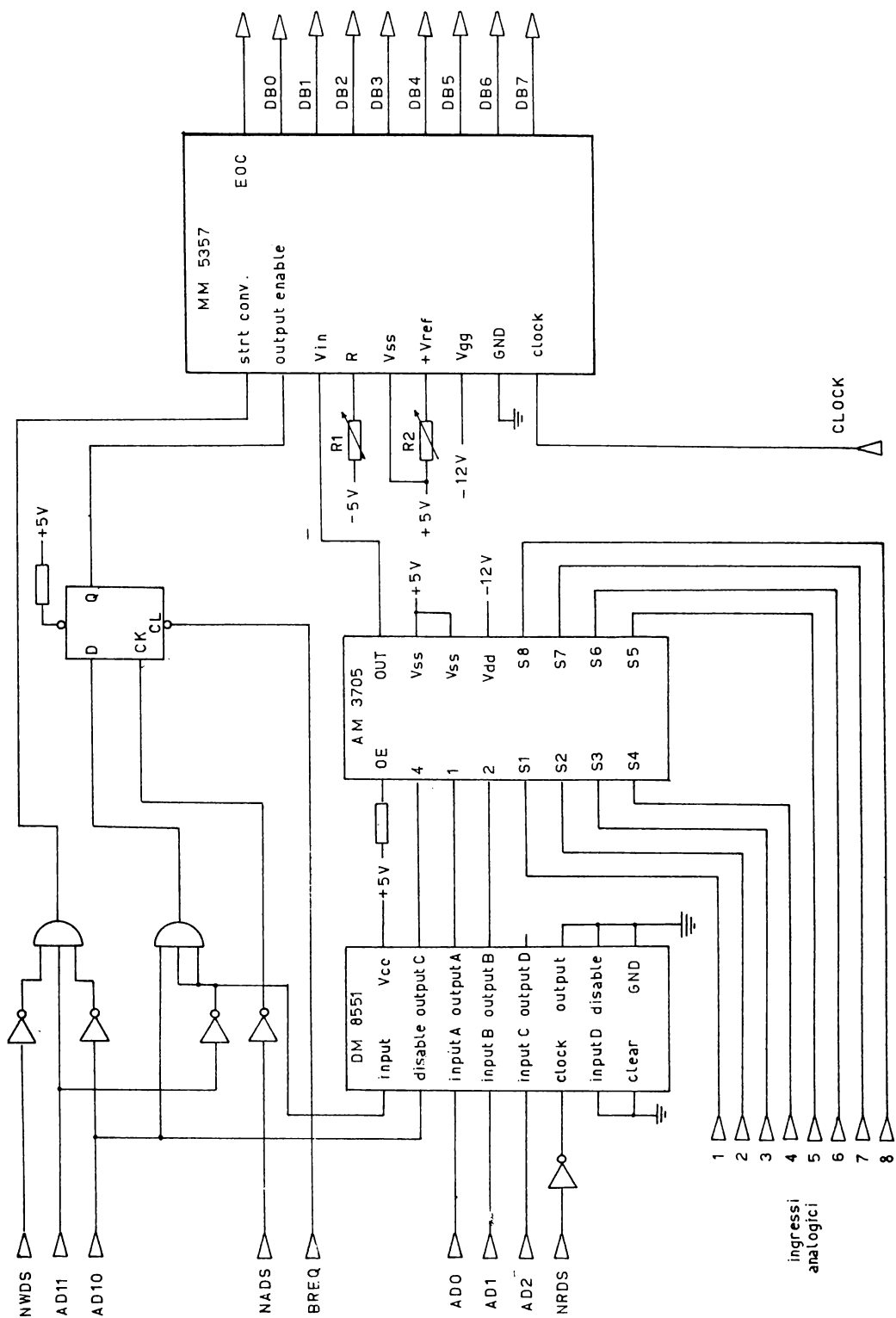



Fig. 1 - 5. Convertitore Analogico/Digitale con multiplexer d'ingresso

Fig. 1 - 6. Conversione Analogico/digitale con multiplexer d'ingresso: (a) listing, (b) flow-chart

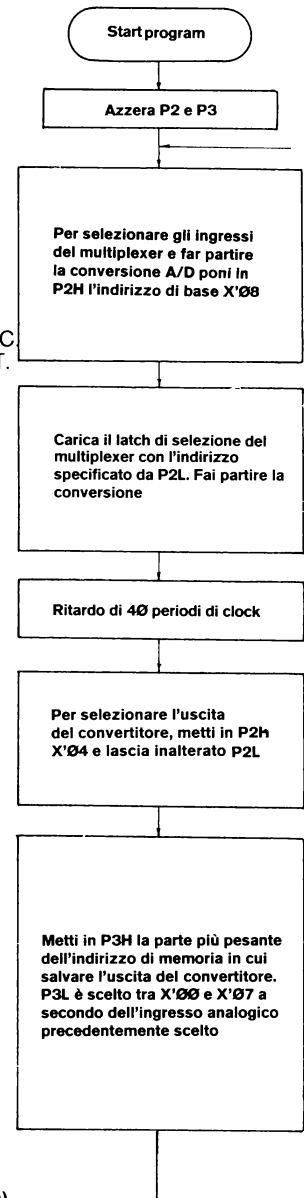
```
NSC SC/MP ASSEMBLER
MEMORY = 0:7

NEXT ASSEMBLY
*.ASM PT

END PASS 1
  1                    ;
  2                         .TITLE SCMP, 'MULTIPLE A/D CNVTR WITH ANALOG MUX'
  3                    ;
  4                    ;
  5         0001            P1      = 1
  6         0002            P2      = 2
  7         0003            P3      = 3
  8                    ;
  9                    ;
 10  0000   C400  START:    LDI     0       ; AZZERA P2L E
 11  0002   32              XPAL    P2      ; P3L
 12  0003   C400            LDI     0
 13  0005   33              XPAL    P3
 14  0006   C408  LOOP:     LDI     08      ; L'INDIR. DELLO SWITCH
 15  0008   36              XPAH    P2      ; ANALOG. ATTUALE È MESSO
 16  0009   C200            LD      (P2)    ; NEL LATCH DI SELEZIONE
 17  000B   CA00            ST      (P2)    ; START ALLA CONVERSIONE
 18  000D   C404            LDI     4
 19  000F   36              XPAH    P2
 20  0010   C402            LDI     2
 21  0012   37              XPAH    P3      ; CONVER. COMPLETA IN 40 MIC
 22  0013   C200            LD      (P2)    ; IN AC USCITA DEL CONVERTIT.
 23  0015   CB00            ST      (P3)    ; MEMORIZZA DATO
 24  0017   33              XPAL    P3
 25  0018   01              XAE
 26  0019   40              LDE
 27  001A   E407            XRI     7       ; VERIF. SE TUTTI I SEGNALI
 28  001C   980A            JZ      EXIT    ; ANAL. SONO CONVERTITI
 29  001E   40              LDE             ; INDIR. DEL PROSSIMO
 30  001F   F401            ADI     1       ; SEGNALE ANALOGICO
 31  0021   01              XAE
 32  0022   40              LDE
 33  0023   33              XPAL    P3
 34  0024   40              LDE
 35  0025   32              XPAL    P2
 36  0026   90DE            JMP     LOOP
 37  0028   08    EXIT:     NOP             ; RITORNO AL PROGRAMMA
 38                                         ; PRINCIPALE
 39                    ;
 40                    ;
 41         0000            .END
SCMP MULTIPLE A/D CNVTR WITH ANALOG MUX.


EXIT    0028      LOOP  0006   P1     0001*
P2      0002      P3    0003   START  0000*


NO ERROR LINES
END PASS 2
SOURCE CHECKSUM = CE80

NEXT ASSEMBLY
*.ASM
```


Fig. 1-6 (a)

1-2 Convertitore Analogico/Digitale con Multiplexer d'ingresso

Descrizione generale

Tranne che per il multiplexer d'ingresso, il convertitore analogico-digitale mostrato in fig. 1-5 è sostanzialmente eguale a quello a singolo ingresso descritto precedentemente (fig. 1-2). Il sistema con multiplexer d'ingresso è evidentemente utile in quei casi in cui sia necessario controllare un numero anche grande di grandezze analogiche ed il costo del sistema sia un fattore determinante riguardo la scelta della soluzione tecnica da adottare.

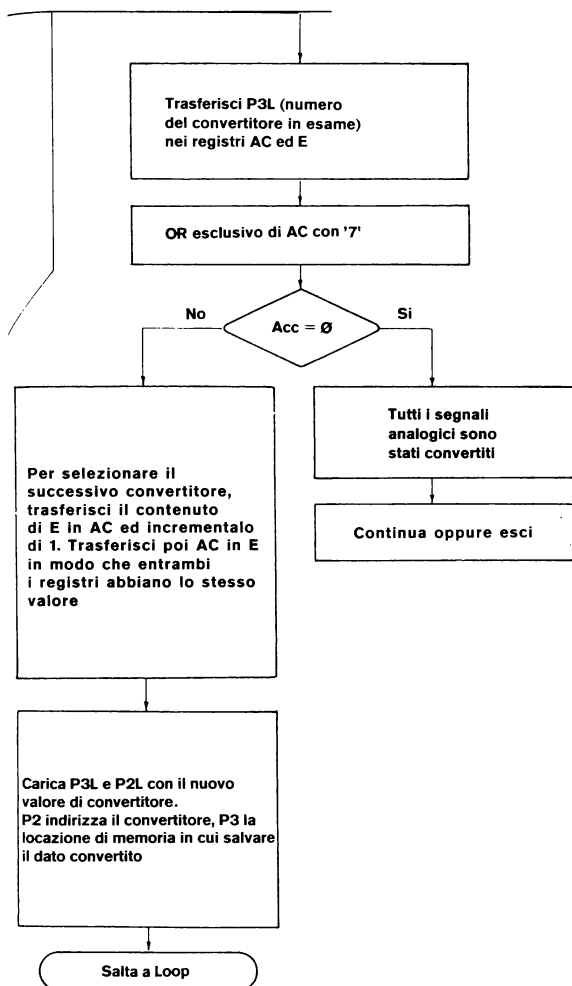



Fig. 1-6 (b)

Operazioni del sistema

Il sistema qui descritto, oltre generare un impulso di start della conversione, deve scegliere su quale ingresso analogico eseguire la conversione stessa. L'indirizzo di start è quindi X'080N, dove N può andare da 0 a 7, selezionando la grandezza analogica che si vuole convertire. L'uscita del convertitore è invece sempre abilitata dall'indirizzo X'0400. L'impulso di start viene generato sostanzialmente dallo strobe NWDS, mentre l'output del convertitore viene abilitato da quando si genera l'impulso di NADS sino a quando il bus dei dati viene liberato dal microprocessore (BREQ diviene basso). L'ingresso analogico da presentare al convertitore viene selezionato tramite un multiplexer ad 8 canali AM3705 che è comandato da un latch. Il latch è realizzato mediante un quad-D flip-flop DM8551 e memorizza il valore N dell'indirizzo di start della conversione quando viene generato un impulso di NRDS (LD all'indirizzo X'080N). Il contenuto del latch decodificato dal multiplexer per scegliere una grandezza analogica delle otto presenti; questo segnale viene quindi presentato al convertitore analogico digitale.

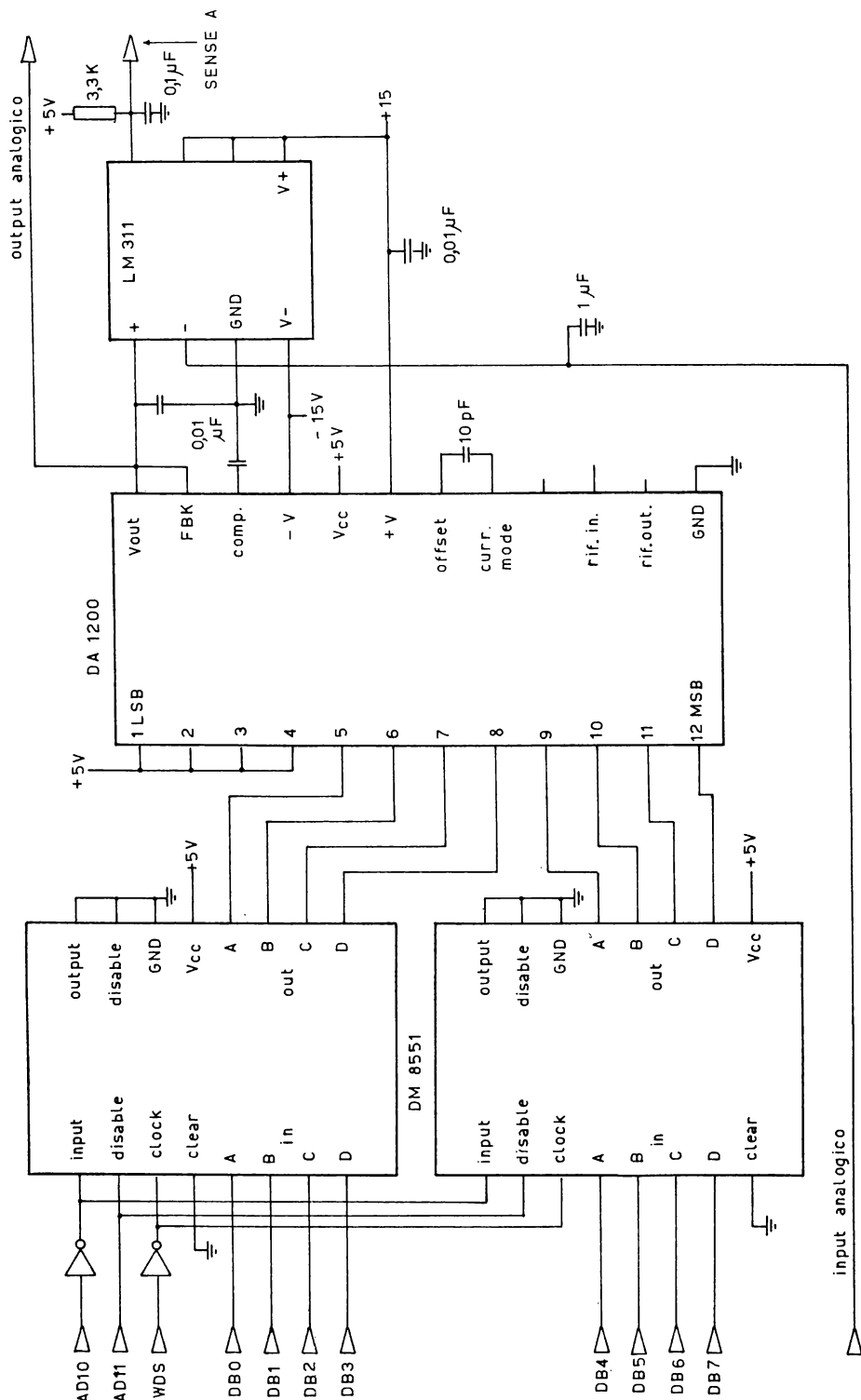
I vari segnali analogici possono quindi essere scelti, sotto controllo software, in modo random oppure sequenzialmente a seconda delle necessità del sistema. Quando una conversione viene completata, il segnale di fine della conversione EOC diventa alto e il risultato della conversione viene memorizzato nel latch di output del convertitore. Le linee di uscita del convertitore sono di tipo tri-state e conseguentemente possono essere connesse direttamente al bus dei dati di SC/MP, permettendo al sistema di essere modulare e flessibile.

Regolazioni da effettuare

La regolazione di zero (tramite R1) e quelle di fondo scala (tramite R2) sono identiche a quelle descritte per il convertitore analogico-digitale a singolo ingresso (fig. 1-2).

Considerazioni software

La fig. 1-6 mostra il flowchart ed il listing del programma che possono essere applicati al sistema qui descritto. Il concetto base è quello di dare lo start alla conversione, attendere che essa sia finita, leggere il risultato della stessa e riporlo in memoria. Il programma è realizzato in modo da eseguire la serie di conversioni nella sequenza da 1 a 8 (indirizzo esadecimale da X'0 a X'7), ma questa è una scelta arbitraria e l'utente può facilmente modificare il tipo di sequenza secondo le necessità del suo sistema modificando opportunamente la sezione meno significativa di P2. Inoltre vi è una corrispondenza univoca tra la grandezza analogica selezionata e la cella in cui viene riposto il suo valore; nel nostro caso il "convertitore" X'0800 viene memorizzato nella cella X'0200, il convertitore X'0801 nella cella X'02' e così via. Dopo ogni conversione il programma testa se tutte le otto conversioni sono state eseguite; come si può notare dal programma, il test viene eseguito mediante l'operazione di exclusive-or tra l'accumulatore ed una costante (in questo caso 7). Se il risultato non è uguale a 0 il programma torna all'inizio per eseguire un'altra conversione, altrimenti il programma torna al programma principale.

Fig. 1 - 7. Convertitore A/D a basso costo

1-3 Conversione Analogico/Digitale a basso costo

Descrizione generale

Il sistema mostrato in fig. 1-7 descrive un metodo semplice e poco costoso per realizzare una conversione digitale-analogica e come è possibile utilizzare il segnale analogico così generato per stabilire il valore di una tensione esterna. Il convertitore analogico-digitale che così si ottiene, richiede la gestione completa della conversione da parte del microprocessore, ma in compenso richiede pochi componenti e pochi segnali di controllo.

Operazioni del sistema

Come mostrato nella figura, due flip-flop operano da latch di ingresso ed hanno gli ingressi connessi al bus dei dati di SC/MP (da DB0 a DB7). Quando il processore presenta un indirizzo opportuno (in questo caso X'0400), i due latch hanno gli ingressi di "data input disable" bassi. Conseguentemente, l'impulso di NWDS del processore carica in essi i dati presenti sul bus (esecuzione dell'istruzione di ST all'indirizzo X'0400). Quindi, gli 8 bit di dato sono memorizzati e presentati al convertitore digitale-analogico (DA 1200), il quale fornirà in uscita l'equivalente analogico del valore digitale presente in ingresso. Data la configurazione di fig. 1-7, se l'ingresso digitale consiste di tutti 1 (FF), l'output analogico sarà di 10 volts, se invece sarà di tutti 0, l'output analogico avrà valore 0 volt. Se ora si suppone di avere come ingresso digitale 10000000 (80), l'output analogico avrà un valore molto vicino a 5 volts; infatti X'80 è praticamente la metà di X'FF. Questa tensione, il cui valore può essere programmato da SC/MP, è quindi direttamente utilizzabile dall'utente per controllare un sistema di tipo analogico. Se però al sistema sopra descritto si aggiunge un comparatore (LM 311), ad un ingresso del quale si connette una tensione esterna (della quale si deve determinare il valore), mentre all'altro ingresso l'output del conver-

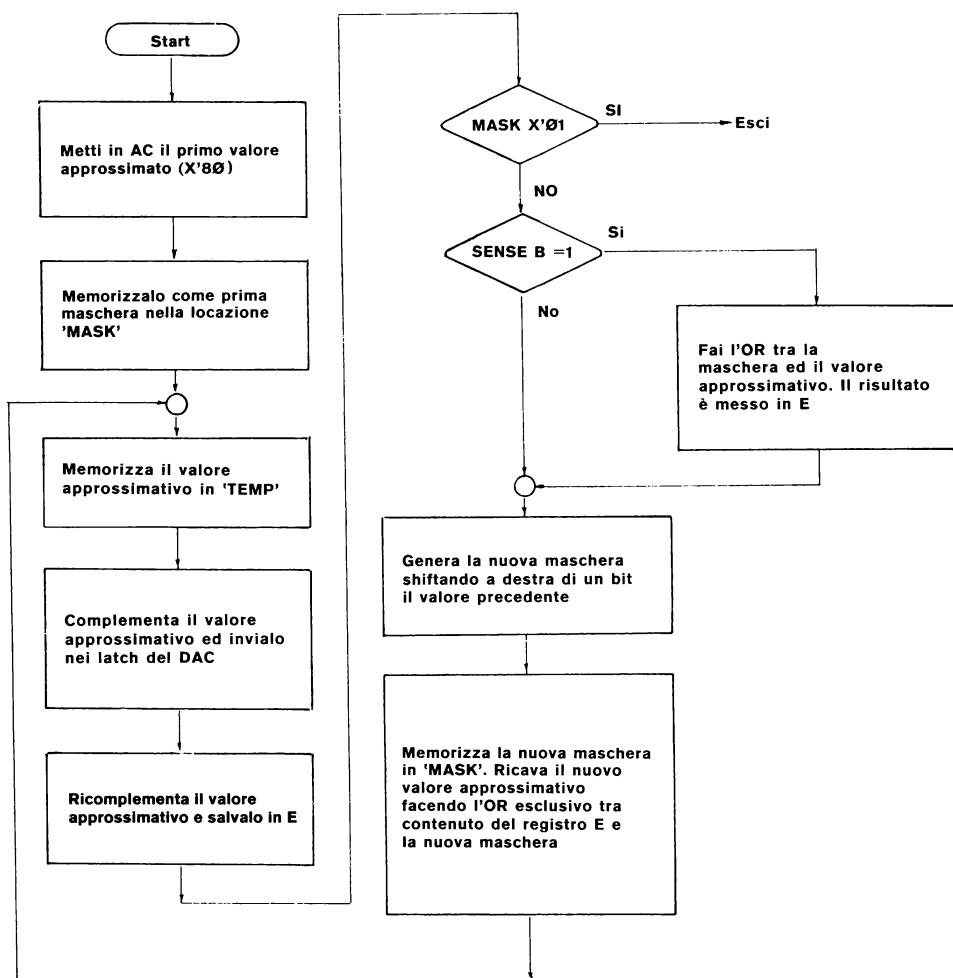



Fig. 1 - 8. Flowchart e listing del programma di conversione A/D a basso costo

titore digitale-analogico del microprocessore SC/MP può, mediante successive approssimazioni, realizzare una conversione analogico-digitale. Il sistema in questione è in grado, in ogni caso, di eseguire una conversione con una risoluzione di 1 su 256, cioè con 8 bit di risoluzione.

Nel caso rappresentato in fig. 1-7, 0 volt è rappresentato da un dato di tutti zero, mentre 10 volts è rappresentato da un dato di tutti uno e quindi il sistema è in grado di eseguire una conversione analogico-digitale di una tensione in ingresso avente lo stesso range di valori limite. Supponiamo ora di avere in ingresso una tensione di 7 volts; il dato in ingresso al convertitore digitale-analogico sarà, per la prima approssimazione, X'80. Poiché la tensione da convertire è maggiore di questo primo valore, la linea di Sense A di SC/MP sarà alta (questa linea è connessa con l'uscita del comparatore e quindi il suo stato indica al processore il risultato del confronto). Il software testa le due condizioni possibili di questa linea (Sense A = "0" o Sense A = "1") in modo tale che se essa è alta deduce che la tensione in ingresso ha un valore compreso tra X'80 e X'FF, mentre se essa è bassa deduce che la tensione in ingresso ha un valore compreso tra X'00 e X'80. Nel nostro caso la tensione in ingresso ha un valore superiore a X'80 ed il processore realizza il secondo test ponendo in ingresso al convertitore digitale-analogico X'C0. A questo punto l'uscita del convertitore ha un valore superiore alla tensione in ingresso e quindi Sense A è posto basso. Viene realizzata quindi una terza approssimazione, ponendo in ingresso al convertitore digitale-analogico X'A0. Il processore continuerà con questo tipo di prove sino a che non avra determinato (con un totale di otto prove) lo stato logico di tutti gli 8 bit della parola di dato equivalente alla tensione in ingresso.
Al termine della conversione, quindi, SC/MP avrà in accumulatore una parola di 8 bit che sarà l'equivalente digitale più vicino possibile alla tensione in ingresso. Questo dato potrà quindi essere posto in memoria e successivamente utilizzato nel modo richiesto dall'applicazione specifica.

Capitolo 2°

INTERFACCIAMENTO DI UNA TASTIERA CON SC/MP

Le applicazioni di SC/MP che richiedono l'interfacciamento con una tastiera possono generalmente utilizzare due metodi per generare il codice del tasto premuto. In un primo metodo, si può utilizzare SC/MP stesso per scandire la tastiera, mentre un altro metodo consiste nel collegare il processore con un keyboard encoder. In entrambi i casi il programma può essere realizzato per eseguire un test continuo della tastiera oppure per utilizzare la tastiera stessa come una periferica gestibile da interruzione. Nelle pagine seguenti verranno presentati alcuni esempi di interfacciamento di SC/MP con una tastiera, applicando le varie ipotesi qui formulate.

2-1 Utilizzo di SC/MP come scanner di tastiera

Descrizione generale

La matrice di tasti mostrata in fig. 2-1 consiste di sei righe aventi ciascuna otto tasti. L'intera tastiera viene scandita testando i dati in ingresso al processore: se il valore di questi dati è diverso da zero, significa che un tasto qualsiasi è stato premuto. Dopo aver stabilito che è stato premuto un qualsiasi tasto, viene realizzata via software una logica antirimbalzo. Quindi, il programma determina quale riga e quale colonna corrispondono al tasto premuto, calcola il codice binario corrispondente, testa il rilascio del tasto, pone il codice del tasto nel registro extension di SC/MP e ritorna al programma principale.

Operazioni del sistema

Alla "periferica tastiera" (keyboard) è stato assegnato l'indirizzo X'0900 e quando SC/MP esegue una istruzione di load (LD) a questo indirizzo, (il buffer tri-state di ingresso viene abilitato). Quindi, se un tasto qualsiasi viene premuto (da S0 a S47), il processore legge sul bus dei dati (da DB0 a DB7) un 1 logico.

Da notare che il buffer che comanda le righe pone le stesse, in questo caso, tutte a 0, mentre il buffer di lettura delle colonne è di tipo invertente, quindi quando viene abilitato ed ha uno 0 logico su un suo ingresso, presenta sulla corrispondenza uscita un 1 logico. Il programma "LOOP" testa questa condizione di "non zero" e provvede alla logica antirimbalzo mediante un'attesa di 5 ms. Dopo queste operazioni, il codice del tasto viene determinato mediante un contatore in RAM. Il contatore viene incrementato di "8" per ogni riga testata e di "1" per ogni colonna. Ad esempio, supponiamo che sia stato premuto il tasto S9 (riga 2/colonna 2 della fig. 2-1); via software, viene testata la prima riga e, non trovando nessun tasto premuto, viene testata la seconda riga, mentre viene incrementato di "8" il contatore di riga. In questa riga, la parola dei dati contiene un 1 logico; conseguentemente, viene ora incrementato di "1" il contatore di colonna e viene confrontato con il dato letto sinché non si ottiene l'eguaglianza. Si può vedere come, tramite i due suddetti contatori, viene generato un codice diverso per ogni tasto premuto. Il codice del tasto viene salvato temporaneamente in memoria e la tastiera viene testata in attesa del rilascio del tasto mediante un'istruzione di load all'indirizzo della tastiera.

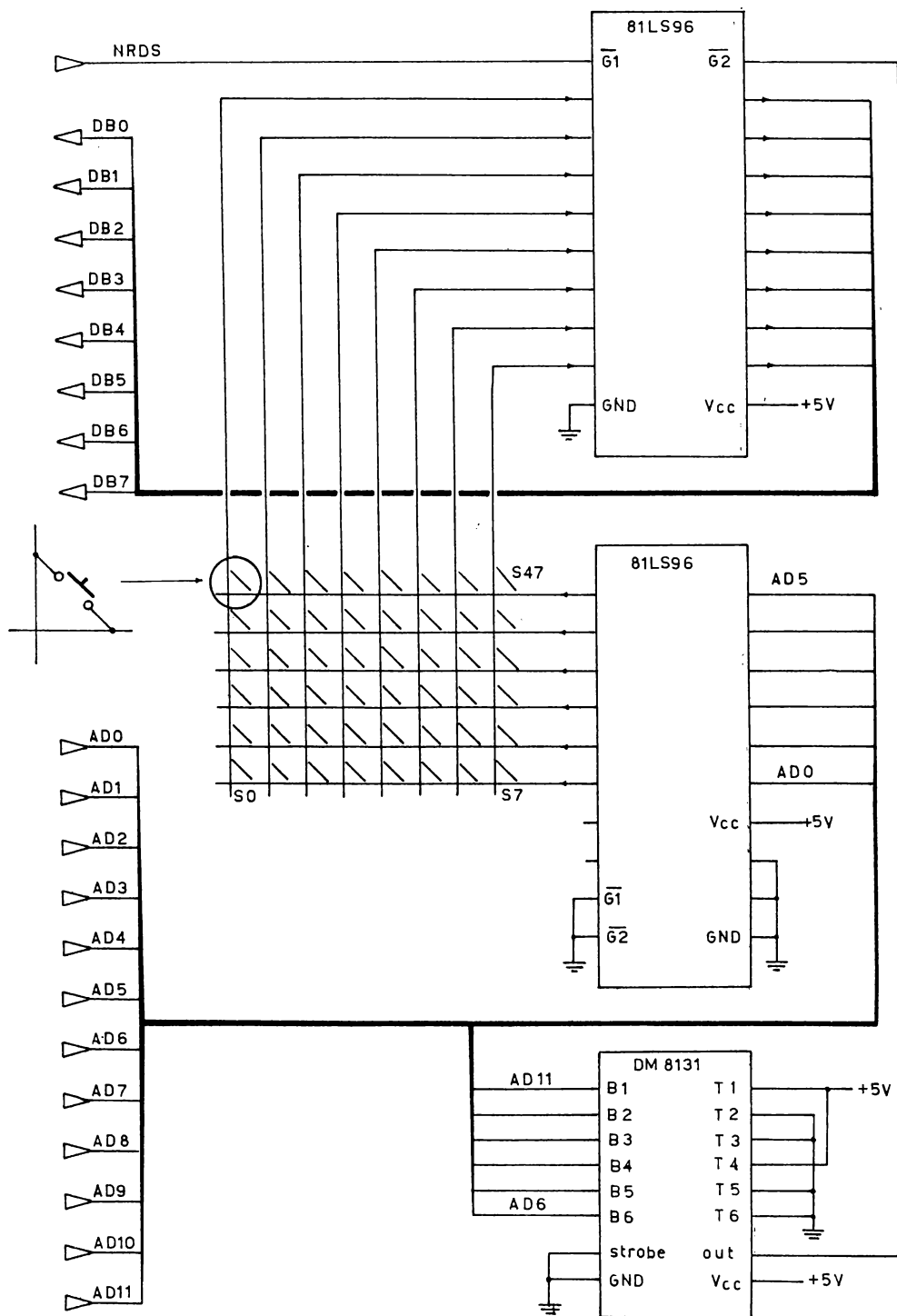



Fig. 2 - 1. Utilizzo di SC/MP come scanner di tastiera

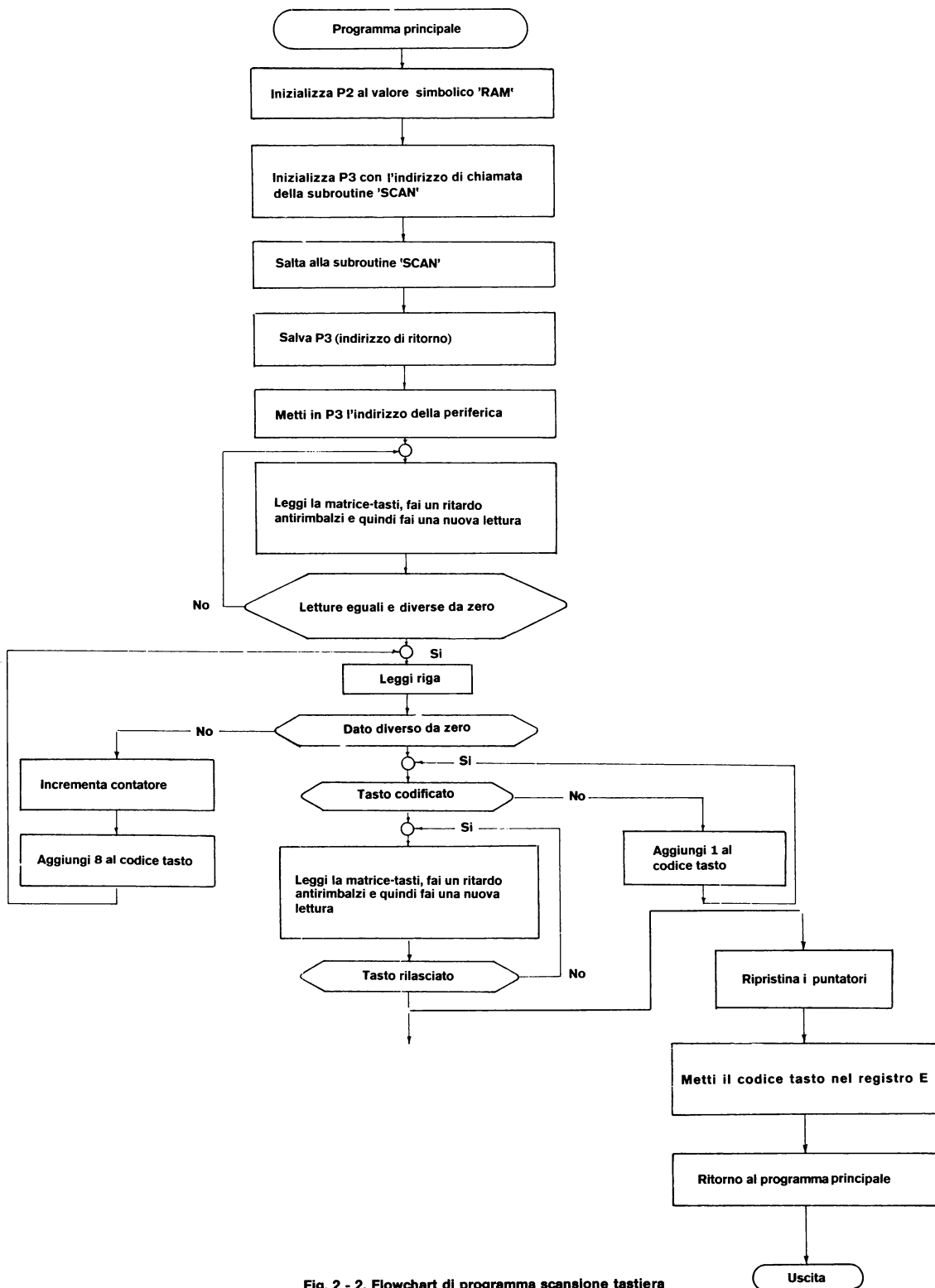



Fig. 2 - 2. Flowchart di programma scansione tastiera

Fig. 2 -3

```
NSC SC/MP ASSEMBLER
MEMORY = 0:7

NEXT ASSEMBLY
*.ASM PT

END PASS 1
  1                        ;
  2                                 . TITLE SCANNR, "SC/MP"
  3                        ;
  4                                 ; SC/MP È UTILIZZATO PER SCANDIRE
  5                                 ; UNA TASTIERA A MATRICE 6 PER 8.
  6                        ;
  7                        ;
  8                                 ; TIPICO PROGRAMMA PRINCIPALE
  9                        ;
 10                        ;
 11                        ;
 12                                 ; QUESTA SEZIONE FISSA IL PUNTATORE "RAM"
 13                                 ; ED IL PUNTATORE DELLA ROUTINE "SCAN".
 14                        ;
 15                        ;
 16  0000  C400                     LDI     L (RAM)
 17  0002  32                       XPAL    2
 18  0003  C403                     LDI     H (RAM)
 19  0005  36                       XPAH    2
 20  0006  C40F                     LDI     L (SCAN) - 1
 21  0008  33                       XPAL    3
 22  0009  C400                     LDI     H (SCAN)
 23  000B  37                       XPAH    3
 24  000C  3F                       XPPC    3              ; CHIAMA ROUTINE "SCAN"
 25  000D  40        START:         LDE                    ; METTI CODICE IN AC.
 26  000E  CA00                     ST      SAVE (2)       ; SALVA CODICE IN "SAVE"
 27                        ;
 28                        ;
 29                                 ; LA ROUTINE "SCAN" FORMA IL CODICE BINARIO
 30                                 ; NELLA CELLA "SWITCH". DOPO AVER VERIFICATO
 31                                 ; IL RILASCIO DEL TASTO, IL CONTROLLO RI-
 32                                 ; TORNA AL PROGRAMMA PRINCIPALE CON IL TASTO
 33                                 ; CODIFICATO NEL REGISTRO E.
 34                        ;
 35                        ;
 36  0010  33        SCAN:          XPAL    3
 37  0011  CA01                     ST      TEMPL (2)      ; SALVA PTR 3
 38  0013  37                       XPAH    3
 39  0014  CA02                     ST      TEMPH (2)
 40  0016  C400                     LDI     L (PERIPH)
 41  0018  33                       XPAL    3
 42  0019  C409                     LDI     H (PERIPH)
 43  001B  37                       XPAH    3              ; INDIRIZZA LA PERIF.
 44                                                        ; COL PUNTATORE P3
 45  001C  02        OVER:          CCL
 46  001D  C400                     LDI     0
 47  001F  CA03                     ST      SWITCH (2)     ; "SWITCH" = 0
 48                        ;
 49                        ;
 50                                 ; LA SCANSIONE DELLA TASTIERA, LA CODIFICA
 51                                 ; DEL TASTO PREMUTO E LA VERIFICA DEL RILASCIO
 52                                 ; INIZIANO CON L'ETICHETTA "LOOP E TERMINA CON
 53                                 ; L'ISTRUZIONE "JNZ RELEAS".
 54                        ;
 55                        ;
 56  0021  C33F      LOOP:          LD      ALLKYS (3)
 57  0023  01                       XAE                    ; SALVA CODICE
 58  0024  8F05                     DLY     5              ; ANTIRIMBALZO  5 MS.
 59  0026  C33F                     LD      ALLKYS (3)
 60  0028  50                       ANE                    ; PARAGONA NUOVO CON VECCHIO
 61  0029  98D5                     JZ      LOOP           ; SE = 0, TASTO NON VAL.
 62  002B  C420                     LDI     X'20
```

segue

Fig. 2 - 3

```
 63 002D 01     LOOP1:  XAE                     ; DRIVER DI RIGA IN AC.
 64 002E C380           LD     - 128 (3)        ; LEGGI RIGA IN AC.
 65 0030 9C0C           JNZ    SHIFT            ; SE DIV. DA 0, VALIDA.
 66 0032 C203           LD     SWITCH (2)
 67 0034 F408           ADI    8                ; INCR. "SWITCH" DI 8
 68 0036 CA03           ST     SWITCH (2)       ;
 69 0038 01             XAE                     ; DRIVER DI RIGA IN AC
 70 0039 1C             SR
 71 003A 98E0           JZ     OVER             ; RICERCA NULLA
 72 003C 90EF           JMP    LOOP1
 73 003E 1C     SHIFT:  SR
 74 003F 9806           JZ     RELEAS           ; SE = 0, TASTO CODIF.
 75 0041 01             XAE                     ; SALVA COD. IN E
 76 0042 AA03           ILD    SWITCH (2)       ; INCREM. "SWITCH"
 77 0044 01             XAE                     ; RICHIAMA COD. PARZIALE
 78 0045 90F7           JMP    SHIFT
 79 0047 C33F   RELEAS: LD     ALLKYS (3)       ; LEGGI MATRICE TASTI
 80 0049 01             XAE                     ; SALVA LETTURA
 81 004A 8F05           DLY    5                ; ANTIRIMBALZO 5 MS.
 82 004C C33F           LD     ALLKYS (3)       ; NUOVA LETTURA
 83 004E 50             ANE                     ; PARAGONA
 84 004F 9CF6           JNZ    RELEAS           ; SE = 0, TASTO RILASC.
 85 0051 C201           LD     TEMPL (2)
 86 0053 33             XPAL   3
 87 0054 C202           LD     TEMPH (2)
 88 0056 37             XPAH   3                ; RIPRISTINA PTR 3
 89 0057 C203           LD     SWITCH (2)       ; TASTO CODIFICATO
 90 0059 01             XAE                     ; SALVA CODICE IN E
 91 005  3F             XPPC   3                ; RITORNO AL PROGRAMMA
 92                                             ; PRINCIPALE
 93 005B 90B3           JMP    SCAN
 94             ;
 95             ;
 96                     ; AREA DEI DATI
 97             ;
 98      0300           RAM    = X'0300
 99      0000           SAVE   = 0
100      0001           TEMPL  = 1
101      0002           TEMPH  = 2
102      0003           SWITCH = 3
103      003F           ALLKYS = X'3F
104      0900           PERIPH = X'0900
105             ;
106             ;
107                     .END
SCANNRSC/MP

ALLKYS  003F    LOOP    0021    TEMPL   0001
LOOP1   002D    OVER    001C    PERIPH  0900
RAM     0300    RELEAS  0047    SAVE    0000
SCAN    0010    SHIFT   003E    START   000D*
SWITCH  0003    TEMPH   0002

END PASS 2
SOURCE CHECKSUM = 4715

NEXT ASSEMBLY
*.ASM
```

Al rilascio del tasto, il codice dello stesso viene prelevato dalla memoria e posto nel registro extension. Il registro puntatore P3 viene scambiato con il program counter PC, eseguendo così il ritorno al programma principale. Il flowchart in fig. 2-2 ed il listing del programma di fig. 2-3 mostrano come è possibile utilizzare SC/MP come scanner di una tastiera.

2-2 Utilizzo di SC/MP con un Keyboard Encoder (20 tasti)

Descrizione generale

La matrice di tasti ed il keyboard encoder mostrati in fig. 2-4 possono essere usati da SC/MP per rea-

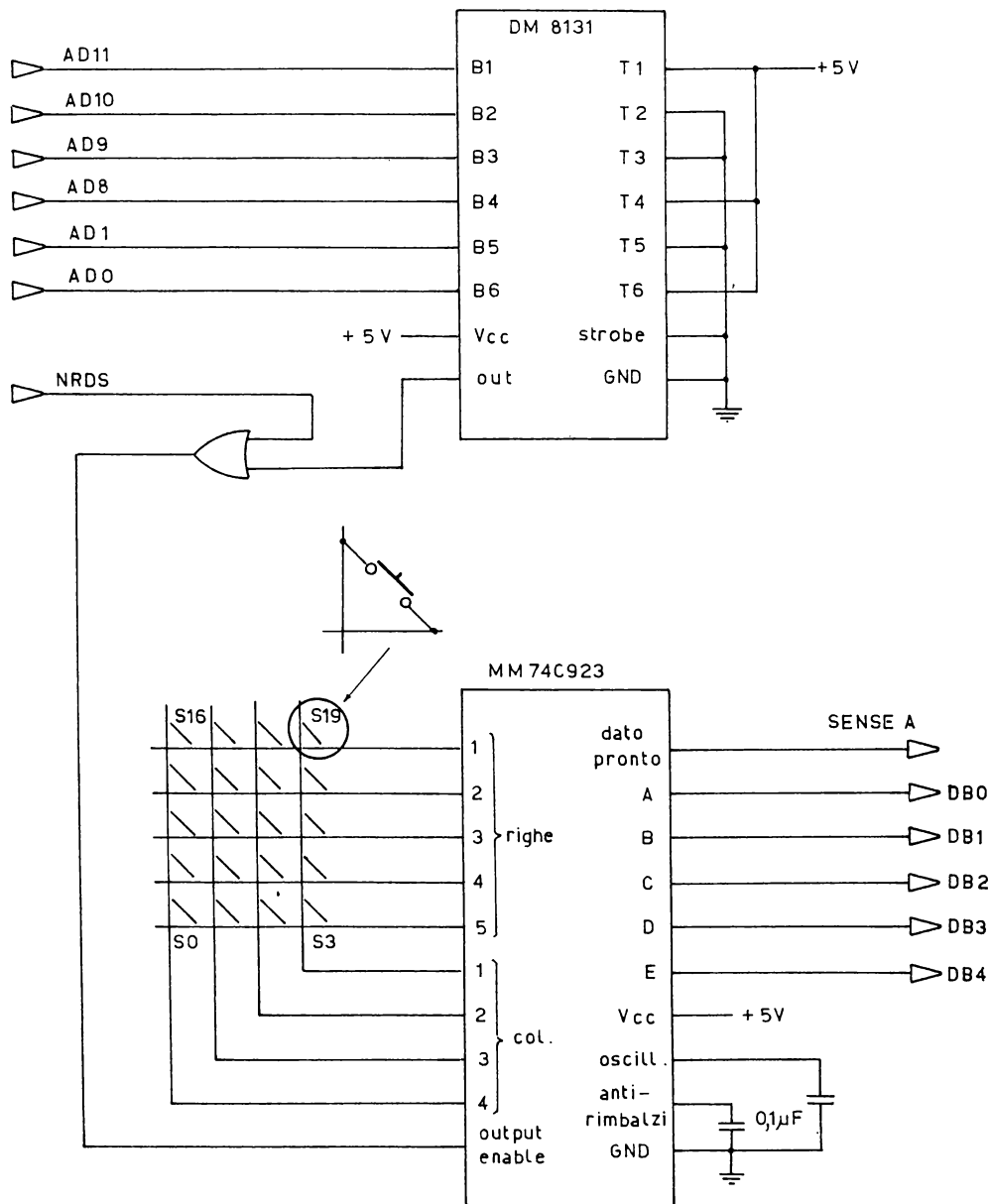



Fig. 2 - 4. Utilizzo di SC/MP con un Keyboard Encoder

lizzare un test continuo della tastiera, oppure per gestire la stessa come periferica in grado di generare un'interruzione. In caso di test continuo, la linea di Sense B di SC/MP viene interrogata continuamente dal processore stesso e quando viene trovata ad "1", i 5 bit di codice del tasto vengono posti in RAM ed il cessore eseguire il programma dell'utilizzatore che gestisce il codice del tasto. Quando invece la tastiera viene utilizzata come una periferica in grado di generare una richiesta di interruzione, il processore esegue un programma "principale" sinché il keyboard encoder non richiesta di interruzione, il processore esegue un programma "principale" sinché il keyboard encoder non pone ad "1" la linea di Sense A. Il programma quindi salta ad una routine di gestione dell'interruzione che legge il codice del tasto, lo pone nel registro extension e ritorna poi al programma interrotto.

Operazioni del sistema

Come mostrato in fig. 2-4, l'encoder CMOS (MM 74C923) possiede la logica necessaria alla codifica dei tasti, alla protezione antirimbalzo ed alla funzione di rollover per due tasti. Quest'ultima funzione garantisce che il segnale di Data Available generato dall'encoder abbia una transizione dallo "0" a "1" logico per ogni tasto premuto, anche nel caso in cui venga premuto un secondo tasto quando il primo non sia stato rilasciato. La segnalazione di Data Available (ed il codice relativo) per quanto riguarda il secondo tasto viene presentata solamente dopo un certo tempo, fissato dalla logica antirimbalzo, dal rilascio del primo tasto. Le operazioni eseguite dal keyboard encoder sono molto semplici, ma efficienti. Quando un tasto viene premuto, l'encoder esegue le operazioni necessarie alla eliminazione dei rimbalzi del contatto, quindi memorizza il codice del tasto premuto in cinque latches tri-state; la presenza di un dato nei suddetti latches viene segnalata dallo stato logico "1" della linea Data Available. Come mostrato, questa linea può essere collegata semplicemente all'ingresso di Sense A di SC/MP. In entrambi i casi, il processore risponde presentando l'indirizzo prefissato del keyboard encoder in coincidenza con il segnale di lettura NRDS. Il processore cioè esegue un'istruzione di load all'indirizzo della tastiera. Conseguentemente, viene comandata bassa (livello logico "0") la linea di output enable del keyboard encoder ed il codice del tasto premuto viene letto dal processore.

Considerazione software

Le fig. 2-5 e 2-6 costituiscono esempi limitati di programmi necessari ad eseguire un test continuo della tastiera o di gestione di un'interruzione generata dall'encoder. In programmi più sofisticati, oltre alla scansione stessa della tastiera. può essere eseguita una certa quantità di controlli sui codici dei tasti premuti, onde assicurare un'elaborazione corretta dei dati introdotti. La fig. 2-5 mostra un programma che utilizza l'ingresso di Sense B per il controllo dell'encoder di tastiera ed il registro extension per contare i caratteri; i codici di questi ultimi sono posti in memoria.
La fig. 2-6 invece utilizza l'ingresso di Sense A come ingresso di richiesta di interruzione; il programma di gestione della interruzione pone il codice del tasto premuto nel registro extension, quindi ritorna al programma principale.

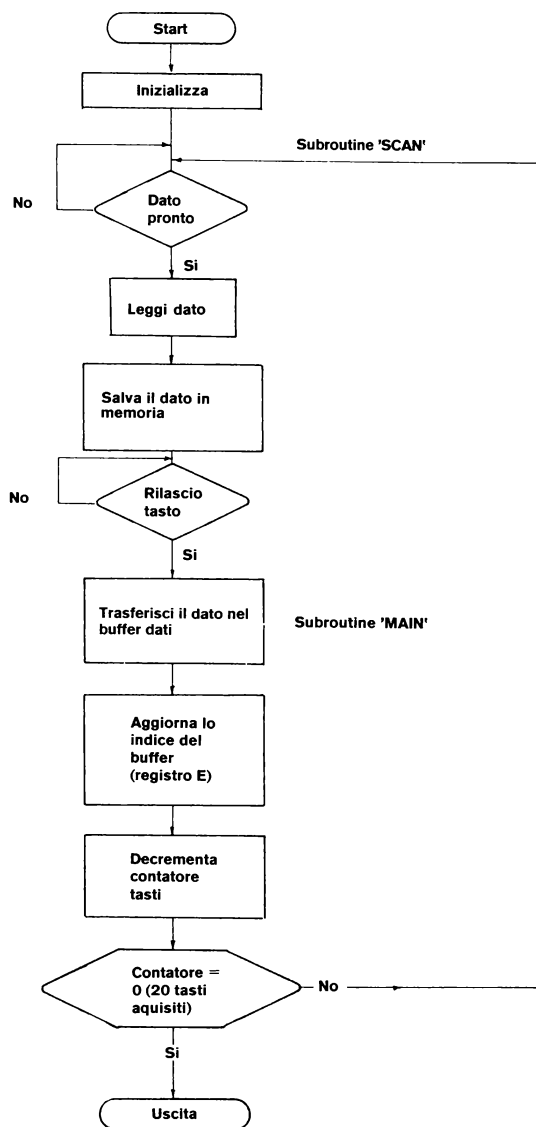



Fig. 2 - 5 (a)

Fig. 2 - 5 Flow chart (a) e listing (b) del programma utilizzante Sense B

Fig. 2 - 5 (b)

```
END PASS 1
  1                              .TITLE SCAN, 20 KEY KYBD SCAN
  2                           ;
  3                           ;
  4                              ; QUESTO PROGRAMMA UTILIZZA UN 74C923
  5                              ; (20 KEY KEYBOARD ENCODER). CON LA
  6                              ; PARTE DI PROGRAMMA ETICHETTATA "MAIN"
  7                              ; SI ACCEDE ALLA ROUTINE "SCAN" DI
  8                              ; TASTIERA.
  9                              ; SENSE B È CONNESSO AL SEGNALE DI "DATA
 10                              ; VALID" DEL 74C923.
 11                              ; IL RITORNO AL PROGRAMMA UTENTE DALLA
 12                              ; ROUTINE "SCAN" AVVIENE DOPO CHE CON
 13                              ; QUESTA SONO STATI PRELEVATI E
 14                              ; MESSI IN MEMORIA 20 TASTI DECODIFICATI.
 15                           ;
 16                           ;
 17  0000  08                    NOP
 18  0001  04                    DINT
 19  0002  C400                  LDI    L (RAM)
 20  0004  32                    XPAL   2
 21  0005  C403                  LDI    H (RAM)
 22  0007  36                    XPAH   2          ; PTR2 = BUFFER DATI
 23  0008  C414                  LDI    X'14       ; MEMORIZ. CONTEGGIO
 24  000A  CA00                  ST     (2)        ; TASTI
 25  000C  05                    IEN               ; ABILITA INTERRUPT
 26  000D  C402                  LDI    ;
 27  000F  01                    XAE               ; REG. E = 2
 28  0010  C401  ENTRY:          LDI    L (KYBD)
 29  0012  31                    XPAL   1
 30  0013  C409                  LDI    H (KYBD)
 31  0015  35                    XPAH   1          ; PTR1 = IND. TASTIERA
 32                        ;
 33                        ;
 34                              ; LE VERIFICHE PER L'INGRESSO DEI
 35                              ; DATI DA TASTIERA E PER IL RILA-
 36                              ; SCIO DEL TASTO INIZIANO CON LA
 37                              ; ETICHETTA "SCAN" E TERMINANO
 38                              ; CON L'ISTRUZIONE "JNZ RELEAS".
 39                        ;
 40                        ;
 41  0016  06    SCAN:           CSA
 42  0017  D420                  ANI    X'20       ; VER. STATO SENSE B
 43  0019  9C02                  JNZ    INPUT      ; SENSE B = 1, TASTO VAL.
 44  001B  90F9                  JMP    SCAN       ; SENSE B = 0, CONTINUA
 45  001D  C100  INPUT:          LD     (1)        ; LEGGI TASTIERA
 46  001F  D40F                  ANI    X'F        ; AZZ. I 3 BIT PIÙ PESANTI
 47  0021  CA01                  ST     1 (2)      ; MEM. CODICE TASTO
 48  0023  06    RELEAS:         CSA
 49  0024  D420                  ANI    X'20       ; VER. STATO SENSE B
 50  0026  9CFB                  JNZ    RELEAS     ; SE ACC. = 1, CONTINUA
 51                                                ; IL LOOP FINCHÉ IL
 52                                                ; TASTO NON È RILASC.
 53  0028  C201  MAIN:           LD     1 (2)      ; PRENDI COD. TASTO
 54  002A  CA80                  ST     - 128 (2)  ; MEM. NEL BUF. DATI
 55  002C  02                    CCL
 56  002D  40                    LDE
 57  002E  F401                  ADI    1
 58  0030  01                    XAE               ; AGGIORNA REG. E
 59                                                ; PER UTILIZZARLO
 60                                                ; COME IND. BUFF. DATI
 61  0031  BA00                  DLD    (2)        ; DECR. CONT. TASTI
 62  0023  9CE1                  JNZ    SCAN
```

segue

Fig. 2 - 5 (b)

```
63  0035   08     EXIT:   NOP               ; DOPO L'ACQUISIZIONE
64                                          ; DI 20 TASTI SI ACCEDE
65                                          ; AL PROGR. UTENTE
66                ;
67                ;
68         0901           KYBD   =X'0901
69         0300           RAM    =X'0300
70                ;
71                ;
72         0000           .END

SCAN 20 KEY KYBD SCAN

ENTRY  0010*      EXIT    0035*      MAIN   0028*
INPUT  001D       KYBD    0901       SCAN   0016
RAM    0300       RELEAS  0023

NO ERROR LINES
END PASS 2
SOURCE CHECKSUM = B8E0

NEXT ASSEMBLY
*.ASM
```

Fig. 2 - 6. Flowchart (a) e listing (b) del programma utilizzante Sense A come richiesta interruzione

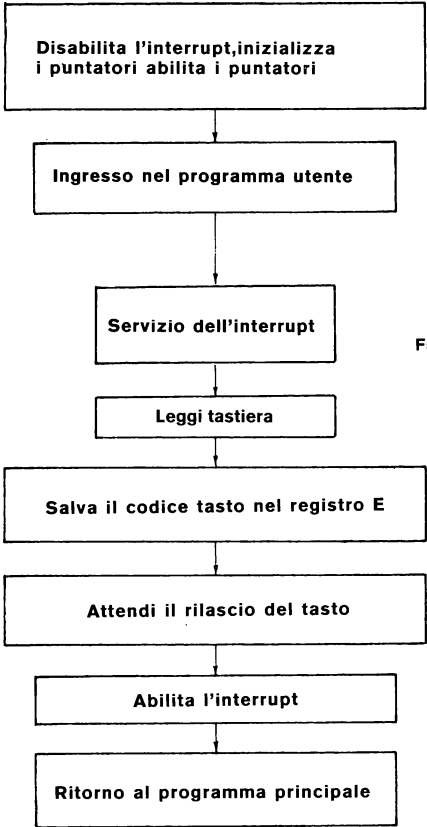



Fig. 2 - 6 (a)

La fig. 2-6 invece utilizza l'ingresso di Sense A come ingresso di richiesta di interruzione; il programma di gestione della interruzione pone il codice del tasto premuto nel registro extension, quindi ritorna al programma principale.

Fig. 2 - 6 (b)

```
NSC SC/MP ASSEMBLER
MEMORY = 0:7

NEXT ASSEMBLY
*.ASM PT, OM

END PASS 1
   1                    ;
   2                            .TITLE SCAN 1, 'SC/MP TO KEY KEYBOARD'
   3                    ;
   4                    ;
   5                            ; QUESTO PROGRAMMA UTILIZZA UN 74C923
   6                            ; (KEYBOARD ENCODER) E SENTE
   7                            ; TRAMITE SENSE A LA RICHIESTA
   8                            ; DI INTERRUPT.
   9                            ; IL CODICE DEL TASTO NON È
  10                            ; ELABORATO
  11                            ; LA DIGITAZIONE DI UN TASTO
  12                            ; CAUSA UN INTERRUPT CHE FORZA IL
  13                            ; PROCESSORE A PRENDERE IL RELATIVO
  14                            ; CODICE.
  15                    ;
  16                    ;
  17                            ; INIZ. ED ATTESA DELL'INTERRUPT
  18                    ;
  19  0000  08                  NOP
  20  0001  04                  DINT                     ; DIS. INTERRUPT
  21  0002  C410                LDI     L (KEYSAV) - 1   ; PTR3 = INDIRIZZO
  22  0004  33                  XPAL    3                ; SERVIZIO INTERRUPT
  23  0005  C400                LDI     H (KEYSAV)
  24  0007  37                  XPAH    3
  25  0008  C401                LDI     L (KYBD)
  26  000A  31                  PAH     1
  27  000B  C409                LDI     H (KYBD)
  28  000D  35                  XPAH    1                ; PTR1 = IND. TASTIERA
  29  000E  05                  IEN                      ; ABILIT. INTERRUPT
  30                    ;
  31                    ;
  32                            ; LA PROSSIMA ISTRUZIONE SIMULA
  33                            ; IL PROGRAMMA UTENTE PRINCIPALE.
  34                            ; IN QUESTO CASO IL PROG. GIRA SULLA
  35                            ; MEDESIMA ISTRUZIONE ('JMP LOOP')
  36                            ; IN ATTESA DELL'INTERRUPT DA TASTIERA.
  37                    ;
  38                    ;
  39  000F  90FE  LOOP:         JMP     LOOP
  40                    ;
  41                    ;
  42                            ; QUELLA CHE SEGUE È LA ROUTINE DI
  43                            ; SERVIZIO DELL'INTERRUPT.
  44                            ; QUESTA ROUTINE PRENDE IL CODICE
  45                            ; DEL TASTO, LO SALVA NEL REGISTRO
  46                            ; E ED INFINE PREDISPONE IL RITORNO
  47                            ; AL PROGRAMMA PRINCIPALE.
  48                    ;
  49                    ;
  50  0011  C100  KEYSAV:       LD      (1)              ; LEGGI COD. TASTO
  51  0013  D41F                ANI     X'1F             ; AZZ. I 3 BIT PIÙ PES.
  52  0015  01                  XAE                      ; SALVA COD. IN E
  53  0016  06    RELEAS:       CSA
  54  0017  D410                ANI     X'10             ; VERIFICA SENSE A
  55  0019  9CFB  JNZ           REL. .A.                 ; ATTENDI RIL. TASTO
  56                    ;
  57                    ;
```

segue

(fig. 2 - 6 (b)

```
58                            ; A QUESTO PUNTO L'UTENTE PUÒ
59                            ; O MEMORIZZARE IN UN BUFFER OPPOR-
60                            ; TUNO DI RAM IL CODICE DEL TASTO
61                            ; OPPURE ELABORARE IL CODICE LETTO
62                            ; NELLA ROUTINE DI INTERRUPT.
63                  ;
64                  ;
65  001B  C702          LD     @2(3)       ; MODIFICA PTR3 PER
66                                         ; SALTARE IL PUNTO
67                                         ; DI LOOP E CONTINUARE IL
68                                         ; PROGR. PRINC.
69  001D  05            IEN                ; ABIL. INTERRUPT
70  001E  3F            XPPC   3           ; RITORNO AL PROGR. PRIN.
71                                         ; CON COD. TASTO IN E
72  001F  90F0          JMP    KEYSAV
73                  ;
74                  ;
75        0901          KEYBD  =X'0901
76                  ;
77        0000          .END

SCAN 1 SC/MP TO KEY KEYBOARD

KEYSAV  0011    KYBD  0901    LOOP  000F
RELEAS  0016

NO ERROR LINES
END PASS 2
SOURCE CHECKSUM = C4A8
TURN PUNCH ON AND HIT ANY KEY
```

2-3 Utilizzo di SC/MP Keyboard Encoder MM5740 (90 tasti)

Descrizione generale

La fig. 2-7 mostra come sia possibile collegare SC/MP con una matrice di tasti abbastanza grande. Questo schema utilizza un encoder per 90 tasti con i circuiti di supporto a lui necessari, un bus comparator per il riconoscimento dell'indirizzo della periferia ed un buffer tri-state di lettura dell'encoder. L'encoder in questione è in grado di presentare un codice di tasto di 9 bit, tuttavia in questo caso sono stati utilizzati solamente 8 bit, 7 per il codice ed 1 per la parità. L'encoder, inoltre, provvede internamente alla logica antirimbalzo ed alla funzione di rollover su 2 (oppure N) tasti.

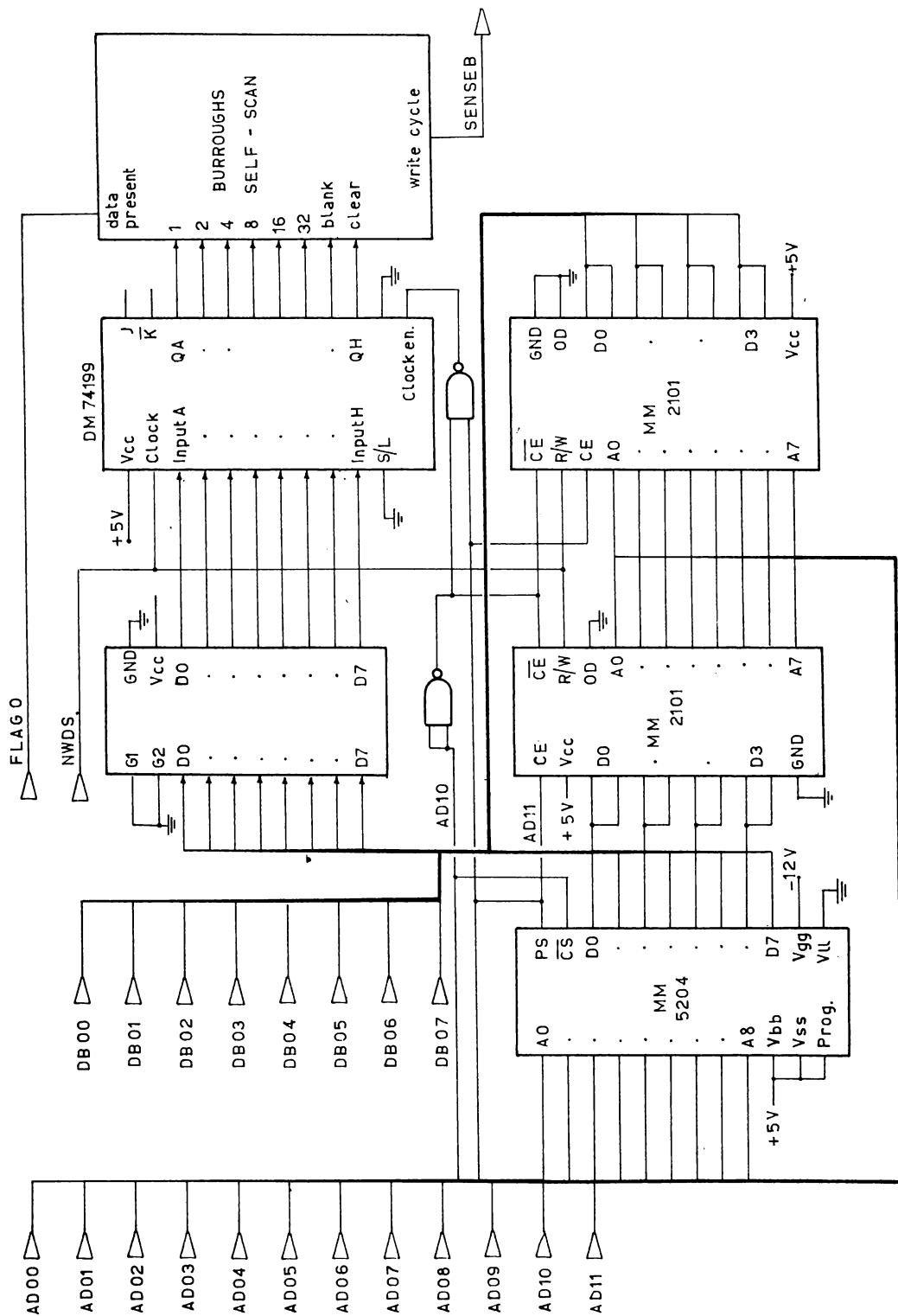
Operazioni del sistema

Il clock necessario all'encoder viene generato tramite un LM 555. Questo circuito è in grado di generare un largo range di frequenze di lavoro. Le funzioni di shift, shift lock e control (generazioni di caratteri speciali di controllo) vengono realizzate tramite i tasti S1, S2, S3; inoltre è previsto l'indicatore della funzione di shift lock. L'encoder opera generando un impulso di strobe dei dati quando questi sono validi; quindi il segnale di "dato pronto" deve essere memorizzato per renderlo compatibile alle necessità di timing della linea di Sense A (o di quella di Sense B) di SC/MP. In ogni caso, il processore risponde presentando l'indirizzo della tastiera al comparatore ed un impulso di lettura NRDS; di conseguenza, il buffer tri-state di lettura viene abilitato ed il codice del tasto premuto viene presentato sul bus dei dati (DB0—DB7) e quindi letto dal processore.

Considerazioni software

Il programma necessario alla gestione del keyboard encoder MM5740 non differisce sostanzialmente da quelli visti precedentemente. In alcune applicazioni, comunque, il medesimo codice può venire interpretato da più programmi. Ad esempio, un programma può realizzare delle funzioni di controllo, mentre un altro può utilizzare gli stessi dati per rappresentarli su di un display mentre un terzo programma può semplicemente utilizzare i dati a scopo statistico.

Fig. 2 - 7. Collegamento di SC/MP con un Keyboard Encoder MM5740

Fig. 2 - 8. Interfaccia tra SC/MP e Display Burroughs Self-Scan

2-4 Interfacciamento di SC/MP con un display Self-Scan (Burroughs)

Descrizione generale

L'interfaccia tra SC/MP ed un display (fig. 2-8) è un esempio di come si possa realizzare un sistema di visualizzazione di un'intera riga di 32 caratteri in modo abbastanza semplice e relativamente poco costoso. Inoltre questo display è in grado di rappresentare 64 tipi di caratteri diversi e quindi il sistema descritto è molto adatto a rappresentare messaggi sia di tipo statico che in movimento.

Operazioni del sistema

Come mostrato in fig. 2-8, i dati provenienti dal bus di SC/MP sono ricevuti da un buffer DM 81LS95 e quindi memorizzati in un latch di tipo DM 74199. Sotto controllo software, il flag 0 viene utilizzato per generare l'impulso di "data present" necessario al display, mentre la linea di Sense B serve al test dello "stato" dello stesso (cioè quando si può presentare al display un nuovo carattere). Ciascuno dei 64 caratteri rappresentabili è definito da un codice di 6 bit codificati in binario: la tabella 2-1 mostra la corrispondenza tra i caratteri ed i loro codici.

Considerazioni software

Il collegamento tra SC/MP e memoria è pure mostrato in fig. 2-8. Il programma di controllo è posto in ROM, dall'indirizzo X'000 al X'01FF; la RAM occupa invece le locazioni dalla X'0F00 alla X'0FFF, mentre il display corrisponde all'indirizzo X'0800. Si deve notare, inoltre che non esistono restrizioni di timing particolari per quanto riguarda l'invio di dati al display self-scan. Ogni carattere del messaggio da scrivere nel display viene prelevato dalla memoria; il programma testa la validità del carattere prelevato e quindi viene eseguita una conversione da codice ASCII a 7 bit a codice ASCII a 6 bit. Dopo questa conversione il bit di clear ed il bit di "blank" del display vengono posti in OR col codice del carattere. Infine, il dato ottenuto viene posto nel latch DM 74199, il processo poi genera un impulso di "data present" (cioè un impulso di scritture nel display) e testa la linea di "write cycle" per conoscere quando il display è pronto ad accettare un nuovo carattere. Le fig. 2-9 e 2-10 mostrano rispettivamente il flowchart ed il listing del programma qui usato per il controllo del display.

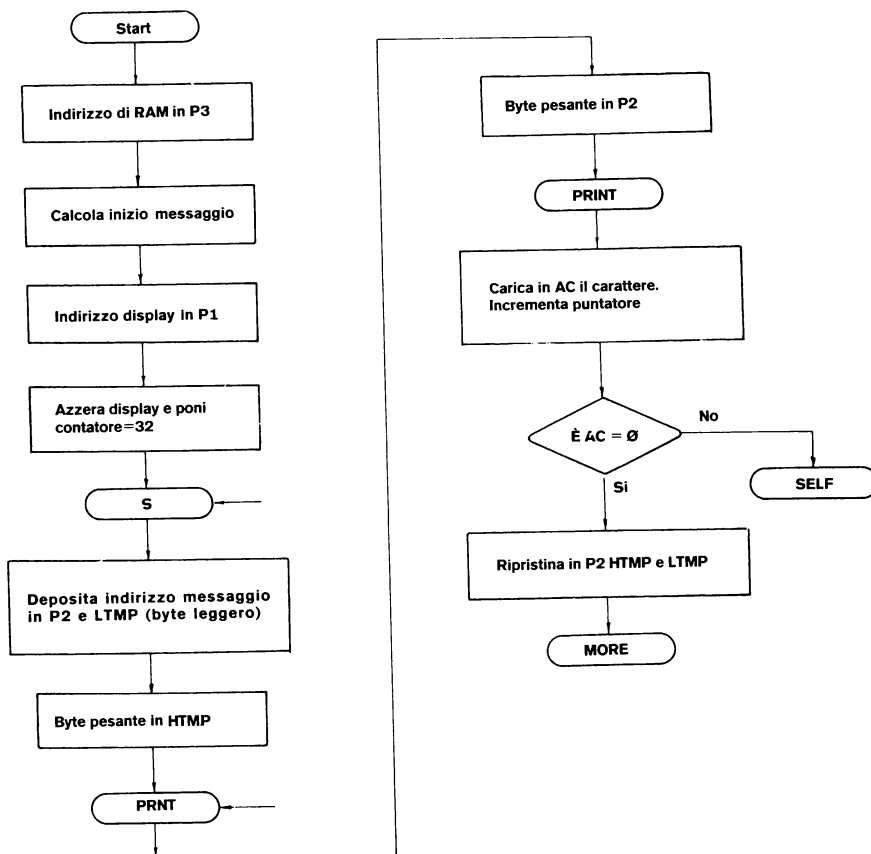



Fig. 2 - 9. Flowchart del programma di controllo Display (Continua)

segue

Tabella 2 - 1. Caratteri alfanumerici e loro codice esadecimale

Codice	Carattere	Codice	Carattere	Codice	Carattere	Codice	Carattere
00	@	10	P	20	(BLANK)	30	0
01	A	11	Q	21	!	31	1
02	B	12	R	22	"	32	2
03	C	13	S	23	#	33	3
04	D	14	T	24	$	34	4
05	E	15	U	25	%	35	5
06	F	16	V	26	&	36	6
07	G	17	W	27	/	37	7
08	H	18	X	28	<	38	8
09	I	19	Y	29	>	39	9
A	J	1A	Z	2A	*	3A	:
B	K	1B	[	2B	+	3B	;
C	L	1C	~	2C	'	3C	<
D	M	1D	]	2D	–	3D	=
E	N	1E	}	2E	.	3E	>
F	O	1F	{	2F	~	3F	?

Fig. 2 - 9.

segue

Fig. 2 - 9.

SELF
Metti AC in E
Ripristina AC
Azzera L e somma
ØEØ ad AC
È AC Ø
No
Si
PRINT
Azzera 1 e somma
ØCØ ad AC
È AC Ø
Si
No
PRINT
Riprendi carattere
da E. Elimina i 2
bits più pesanti
PRINT
OR con Ø8Ø
Invia carattere sul
display
Copia status in AC
Setta flag Ø
Resetta flag Ø
DOWAIT
Copia status in AC.
Maschera AC tranne
in Sense B
AC = 0
No
Si
COUNT
COUNT - 1
COUNT = Ø
No
Si
NEW

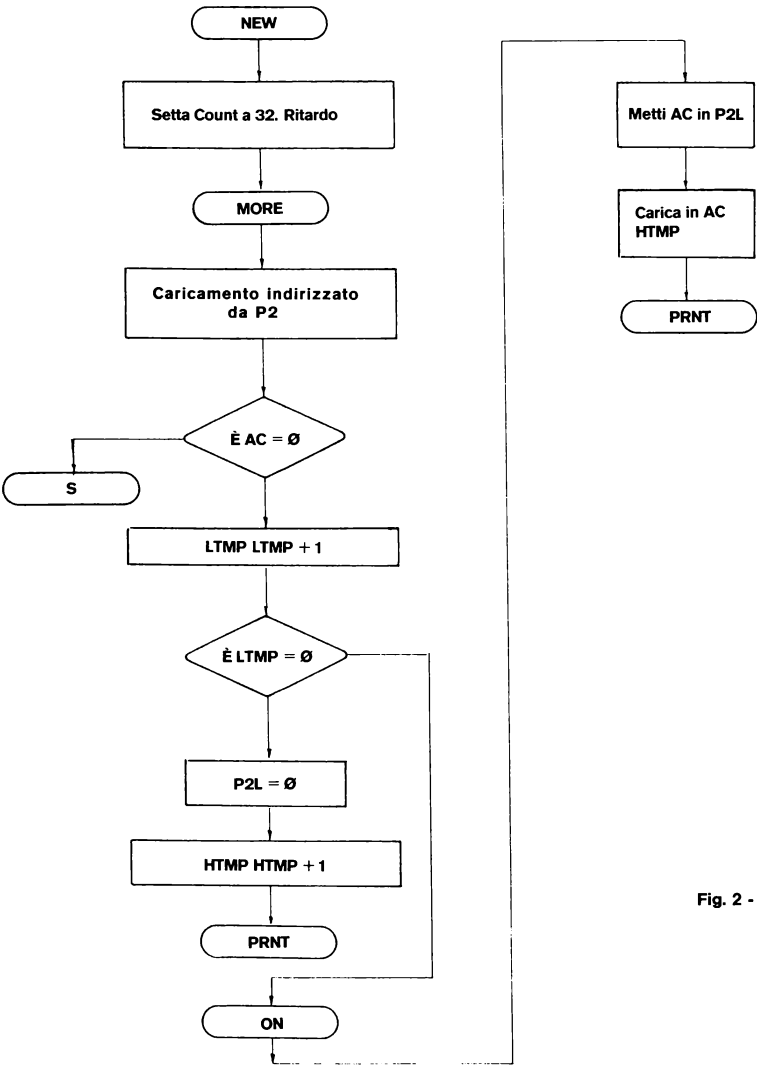
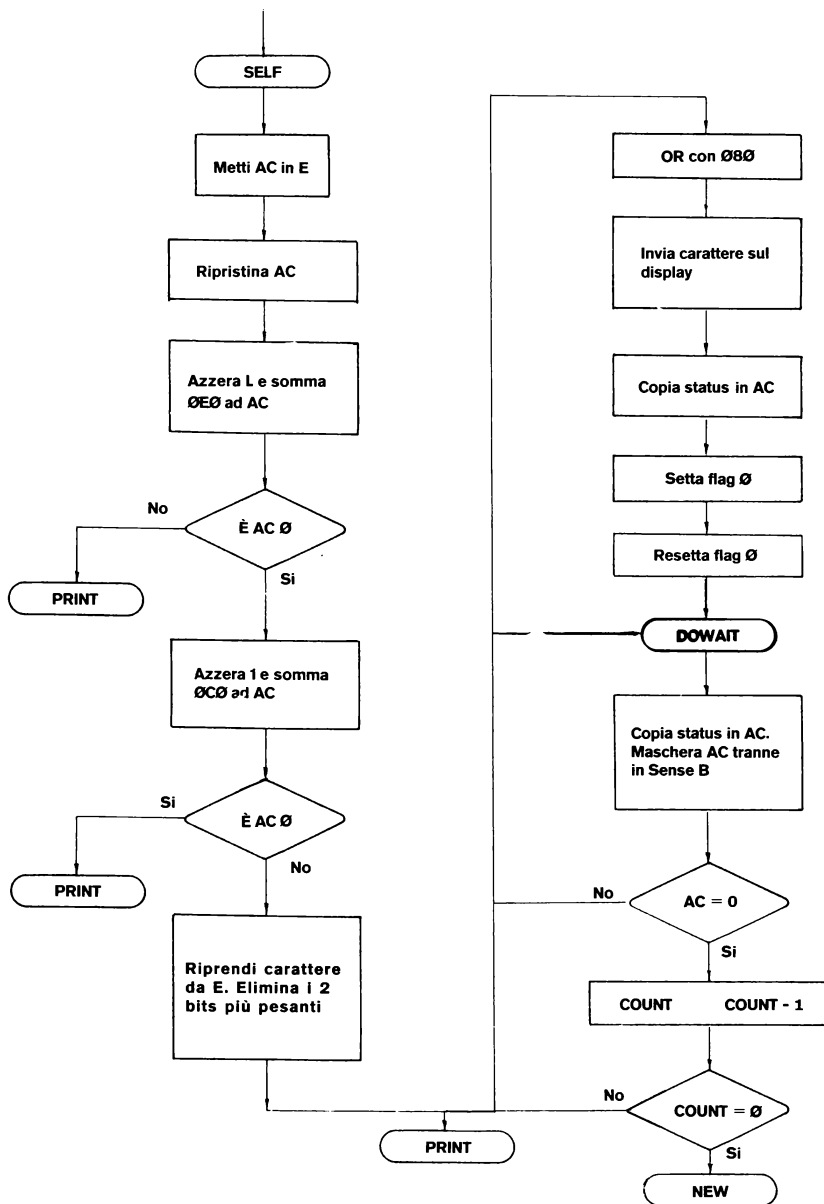



Fig. 2 - 10 Listing del programma di controllo display

```
END PASS 1
   1                          ;
   2                          .TITLE DISP, 'MOVING MESSAGE'
   3                 ;
   4                          ; IL MESSAGGIO DEVE ESSERE > DI 32 CARAT
   5                 ;
   6                 ;
   7                          ; CELLE RAM UTILIZZATE
   8                 ;
   9                 ;
  10                          ; 0F00  INDIRIZZO MESSAGGIO (BYTE PES.)
  11                          ; 0F01  INDIRIZZO MESSAGGIO (BYTE LEG.)
  12                          ; 0F02  SUCCESSIVO IND. MES. (BYTE LEG.)
  13                          ; 0F03  SUCCESSIVO IND. MES. (BYTE PES.)
  14                          ; 0F04  CONTAT. CARAT. DI LINEA
  15                 ;
  16                 ;
  17  0000                    HST    =0        ; COME 0F00
  18  0001                    LST    =1        ; COME 0F01
  19  0002                    HTMP   =2        ; COME 0F02
  20  0003                    LTMP   =3        ; COME 0F03
  21  0004                    COUNT  =4        ; COME 0F04
  22  0800                    ADR    =0800     ; INDIRIZZO DISPLAY
  23  0F00                    RAM    =0F00     ; IND. INIZIO RAM
  24  0000                           .=0       ; IND. INIZIO PROGR.
  25                 ;
  26                 ;
  27                          ; IL MESSAGGIO È IN MEMORIA IN
  28                          ; STRINGHE DI CARATTERI ASCII.
  29                          ; LA FINE DEL MESSAGGIO È INDI-
  30                          ; VIDUATA DA UN BYTE DI TUTTI 0.
  31                 ;
  32                 ;
  33  0F20                    MMSG   =0F20     ; IND. MESSAGGIO
  34                 ;
  35                 ;
  36

DISP MOVING MESSAGE

                              . PAGE
  37                 ;
  38  0000  08      START:    NOP
  39  0001  C40F              LDI    H (RAM)   ; IN P3 IND. RAM
  40  0003  37                XPAH   3
  41  0004  C400              LDI    L (RAM)
  42  0006  33                XPAL   3
  43  0007  C40F              LDI    H (MMSG)  ; MEMORIZZA IND.
  44  0009  CB00              ST     HST (3)   ; INIZIO MES.
  45  000B  C420              LDI    L (MMSG)
  46  000D  CB01              ST     LST (3)
  47  000F  C408              LDI    H (ADR)   ; IN P1 IND. DISPLAY
  48  0011  35                XPAH   1
  49  0012  C400              LDI    L (ADR)
  50  0014  31                XPAL   1
  51  0015  C400              LDI    0         ; AZZERA DISPLAY
  52  0017  C900              ST     (1)
  53  0019  C420              LDI    32        ; CONT. CARAT. = 32
  54  001B  CB04              ST     COUNT (3)
  55                 ;
  56  001D  C301    S:        LD     LST (3)   ; MEMOR. IND. MES.
  57  001F  CB03              ST     LTMP (3)  ; IN LTMP E
  58  0021  32                XPAL   2         ; P2 (BYTE LEGGERO)
  59  0022  C300              LD     HST (3)   ; MEM. IND. MES.
  60  0024  CB02              ST     HTMP (3)  ; IN HTMP E
  61  0026  36      PRNT:     XPAH   2         ; P2 (BYTE PESANTE)
  62  0027  C601    PRINT:    LD     @1 (2)    ; IN AC CARAT. SUCCES.
  63  0029  9C22              JNZ    SELF      ; TEST FINE MES.
  64  002B  C302              LD     HTMP (3)  ; RIPRISTINA PUNTAT.
```

segue

Fig. 2 - 10

```
 65  002D  36               XPAH    2
 66  002E  C303             LD      LTMP (3)
 67  0030  32               XPAL    2
 68  0031  900D             JMP     MORE
 69                 ;
 70  0033  32       ON:     XPAL    2           ; IN P2 BYTE LEGGERO
 71  0034  C302             LD      HTMP (3)    ; METTI IN AC BYTE PES.
 72  0036  90EE             JMP     PRNT
 73                 ;
 74  0038  C420     NEW:    LDI     32
 75  003A  CB04             ST      COUNT (3)
 76  003C  C4FF             LDI     0FF         ; PICCOLO RITARDO
 77  003E  8F80             DLY     080
 78  0040  C200     MORE:   LD      (2)
 79  0042  98D9             JZ      S           ; TEST SE AC = 0
 80  0044  AB03             ILD     LTMP (3)    ; INCREM. LTMP
 81  0046  9CEB             JNZ     ON
 82  0048  32               XPAL    2           ; IND. SUCCES. IN P2L
 83  0049  AB02             ILD     HTMP (3)    ; INCREM. HTMP
 84  004B  90DA             JMP     PRINT
 85                 ;
 86  004D     01     SELF:     XAE                  ; SALVA CAPAT.
 87  004E     40               LDE                  ; CARAT. IN AC
 88  004F     02               CCL                  ; AZZERA LINK
 89  0050     F4E0             ADI     0E0          ; VERIF. SE < 020
 90  0052     9402             JP      GT1F         ; NO
 91  0054     90D1             JMP     PRINT        ; SI. RITORNA
 92  0056     90D1   GT1F:     CCL                  ; AZZERA LINK
 93  0057     F4C0             ADI     0C0          ; VERIF. SE > 05F
 94  0059     94CC             JP      PRINT        ; SI. RITORNA
 95  005B     40               LDE                  ; IN AC CARAT. VALIDO
 96  005C     D43F             ANI     03F          ; ELIMINA BITS PESAN.
 97  005E     DC80             ORI     080
 98  0060     C900             ST      (1)          ; INVIA PAROLA AL DIS.
 99  0062     06               CSA                  ; STATUS IN AC
100  0063     DC01             ORI     1            ; CICLO DI SCRITTURA
101  0065     07               CAS                  ; FLAG 0 = 1
102  0066     D4FE             ANI     0FE
103  0068     07               CAS                  ; FLAG 0 = 0
104  0069     06     DOWAIT:   CSA                  ; STATUS IN AC
105  006A     D420             ANI     020          ; EVIDENZIA SENSE B
106  006C     9CFB             JNZ     DOWAIT       ; SE = 1 RICICLO
107  006E     BB04             DLD     COUNT (3)    ; DECREM. CONT.
108  0070     98C6             JZ      NEW
109  0072     90B3             JMP     PRINT
110                      ;
111                      ;
112           0000             .END

ADR      0800      COTNT   0004*      COUNT   0004
DOWAIT   0069      GT1F    0056       HST     0000
HTMP     0002      LST     0001       LTMP    0003
MMSG     0F20      MORE    0040       NEW     0038
ON       0033      PRINT   0027       PRNT    0026
RAM      0F00      S       001D       SELF    004D
START    0000*

NO ERROR LINES
END PASS 2
SOURCE CHECKSUM = 78CD

NEXT ASSEMBLY
*.ASM
```

Capitolo 3°

INTERFACCIAMENTO DI SC/MP CON CASSETTA MAGNETICA TIPO AUDIO

Descrizione generale

La fig. 3-1 rappresenta lo schema a blocchi di un possibile sistema per interfacciare il microprocessore SC/MP con una cassetta magnetica di tipo audio. Questo metodo di memorizzazione di dati è poco costoso, ma affidabile ed è quindi alternativo a molti altri sistemi di tipo classico che utilizzano mediamente metodi e supporti più complessi e costosi. In questo caso, il registratore può costare da L. 50.000 a L. 100.000, fornendo delle prestazioni decisamente affidabili, mentre d'altra parte una cassetta magnetica da 30 minuti di durata permette la memorizzazione di circa 40K byte di dati per lato. La velocità di scrittura e lettura corrisponde a 330 baud (cioè a circa 40 byte al secondo) ed è quindi sufficiente in molti tipi di applicazioni.

Operazioni del sistema

In questa applicazione si suppone che il sistema SC/MP al quale è connessa l'interfaccia con cassetta magnetica in questione sia il sistema di sviluppo LCDS della National Semiconductor, oppure che il sistema dell'utente preveda la possibilità di trasferire il controllo di SC/MP su comando dell'operatore. Il programma di input/output residente in PROM provvede a tutte le funzioni di controllo e di temporizzazione necessarie per trasmettere e ricevere dati da e per il registratore. Le operazioni da eseguire per trasmettere un blocco di dati possono essere riassunte come segue:

1 - L'operatore prepara il blocco di dati in memoria; questa operazione potrebbe essere compiuta, ad esempio, mediante una tastiera.
2 - Il registratore viene acceso e posto in operazione di scrittura, quindi l'operatore trasferisce il controllo di SC/MP alla routine di scrittura che trasferisce il blocco di dati alla cassetta magnetica tramite l'interfaccia. I led indicatori di stato daranno all'operatore la segnalazione di quando la trasmissione è terminata.

Le operazione da eseguire per la ricezione sono le seguenti:

1 - L'operatore trasferisce il controllo di SC/MP alla routine di ricezione, la quale è in grado di porre in memoria (a partire dall'indirizzo indicato) i dati ricevuti dal registratore.
2 - Il registratore viene acceso e posto in operazione di lettura, quindi l'operatore deve osservare i led indicatori per determinare quando la trasmissione è terminata e se i dati ricevuti sono corretti.

Trasmissione

La trasmissione di dati da SC/MP al registratore è realizzata sulla base di un sistema di sincronizzazione automatica sul singolo bit, mediante il quale si evita la possibilità di errori cumulativi di lettura o scrittura. Il diagramma dei tempi relativo alla trasmissione è mostrato in fig. 3-2. La routine di trasmissione stabilisce un periodo tra gli impulsi di clock di 4ms. Il tempo tra un impulso e l'altro di clock è il tempo di trasmissione del dato, cioè se si deve trasmettere uno "0" non vi sarà un ulteriore impulso tra i due di clock, mentre se si deve trasmettere un "1" logico verrà generato un impulso tra i due predetti di clock. Gli impulsi di clock e di dato vengono generati dalla decodifica degli indirizzi mostrata in fig. 3-2. L'indirizzo è identico, come si può notare, e quindi per generare sia l'impulso di clock che quello di dato, il microprocessore SC/MP deve eseguire un'istruzione di Store al medesimo indirizzo (X'8304). Gli impulsi di clock o di dato vengono quindi inviati lungo la medesima linea al registratore tramite il circuito di interfaccia. Viene quindi generata una sequenza di impulsi negativi, cioè gli impulsi di clock, mentre i dati consistono nella presenza o meno di un impulso tra i due di clock; se si deve trasmettere un "1" logico SC/MP genera un impulso esattamente a metà del periodo tra i due impulsi di clock, se si deve trasmettere uno "0" SC/MP non eseguirà l'istruzione di Store (all'indirizzo X'8304) a metà periodo di clock. Per memorizzare su un nastro magnetico un blocco di dati, la routine di scrittura innanzitutto genera una lunga serie di byte di valore zero (leader) più una parola di identificazione (X'A5).

Come abbiamo visto, questi dati vengono generati dalla decodifica degli indirizzi, la quale a sua volta li presenta al circuito di interfaccia che adatta la durata degli impulsi ed i livelli di tensione a valori accettabili dall'ingresso di microfono del registratore. Vengono poi trasmessi i dati seguiti dalla parola di identificazione X'A5; il formato completo dei dati trasmessi e quindi scritti sulla cassetta sono mostrati in fig. 3-1.

Lettura

Durante le operazioni di ricezione, il processore testa la linea di Sense B in attesa che essa sia allo "0" logico (questa linea è infatti comandata dal latch di ricezione). Quando la linea di Sense B è bassa, significa che è stato ricevuto il primo impulso di clock. Il latch viene quindi resettato dal processore, il quale poi genera un ritardo di metà periodo di clock. Dopo questo ritardo, SC/MP ritorna a testare la linea di Sense B; se essa diviene di nuovo bassa significa che è stato ricevuto un "1" logico dalla cassetta, altrimenti significa che il bit trasmesso è uno "0". Dopo il test sull'informazione relativa al dato trasmesso, il processore torna a testare la linea di Sense B in attesa del secondo clock e quindi del nuovo bit di dato. Ci si riferisce alla fig. 3-2 per i timing relativi alla fase di lettura. Durante la fase di lettura, la routine di ricezione, cerca innanzitutto la parola di identificazione, quindi legge il blocco di dati e li pone in memoria RAM. Il programma è tale per cui, dopo il caricamento dei dati, il controllo viene dato ad un altro programma (ad esempio, un programma principale di debug). L'indirizzo di partenza di questo programma principale viene dedotto dall'Entry Point Address presente nel blocco dei dati letti dalla cassetta magnetica.

Considerazioni software

Per scrivere un blocco di dati nella cassetta magnetica, l'operatore deve caricare nelle locazioni di memoria X'8203 (byte più significativo) e X'8204 (byte meno significativo) l'indirizzo (quattro digit) esadecimale del blocco di dati da trasmettere. Quindi, le locazioni X'820C (byte più significativo) e X'820D (byte meno significativo) devono essere caricate con l'indirizzo di Entry Point eseguito da SC/MP ogni volta che caricherà il suddetto blocco di dati).

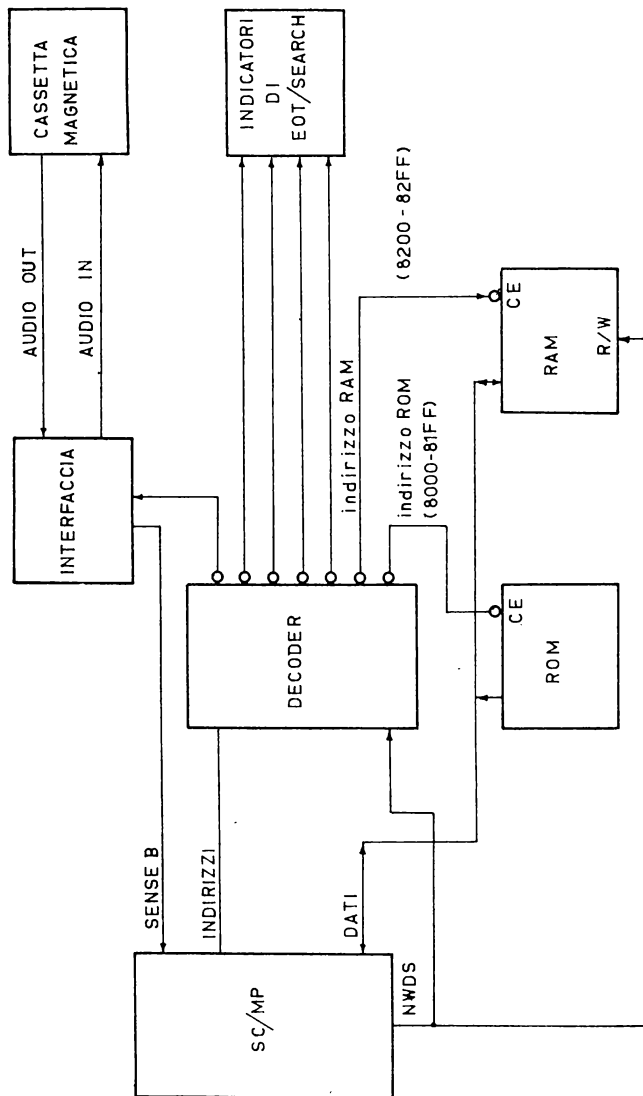
Fig. 3 - 1. Schema a blocchi di interfaccia tra SC/MP e cassetta audio

Leader	Parola di identificazione	Indirizzo di partenza	Entry point	Lunghezza del blocco	Blocco dati	Checksum
128 bytes di 0	X' A5	16 bit	16 bit	16 bit		8 bit

Fig. 3 - 2. Schema di interfaccia tra SC/MP e cassetta audio (continua)

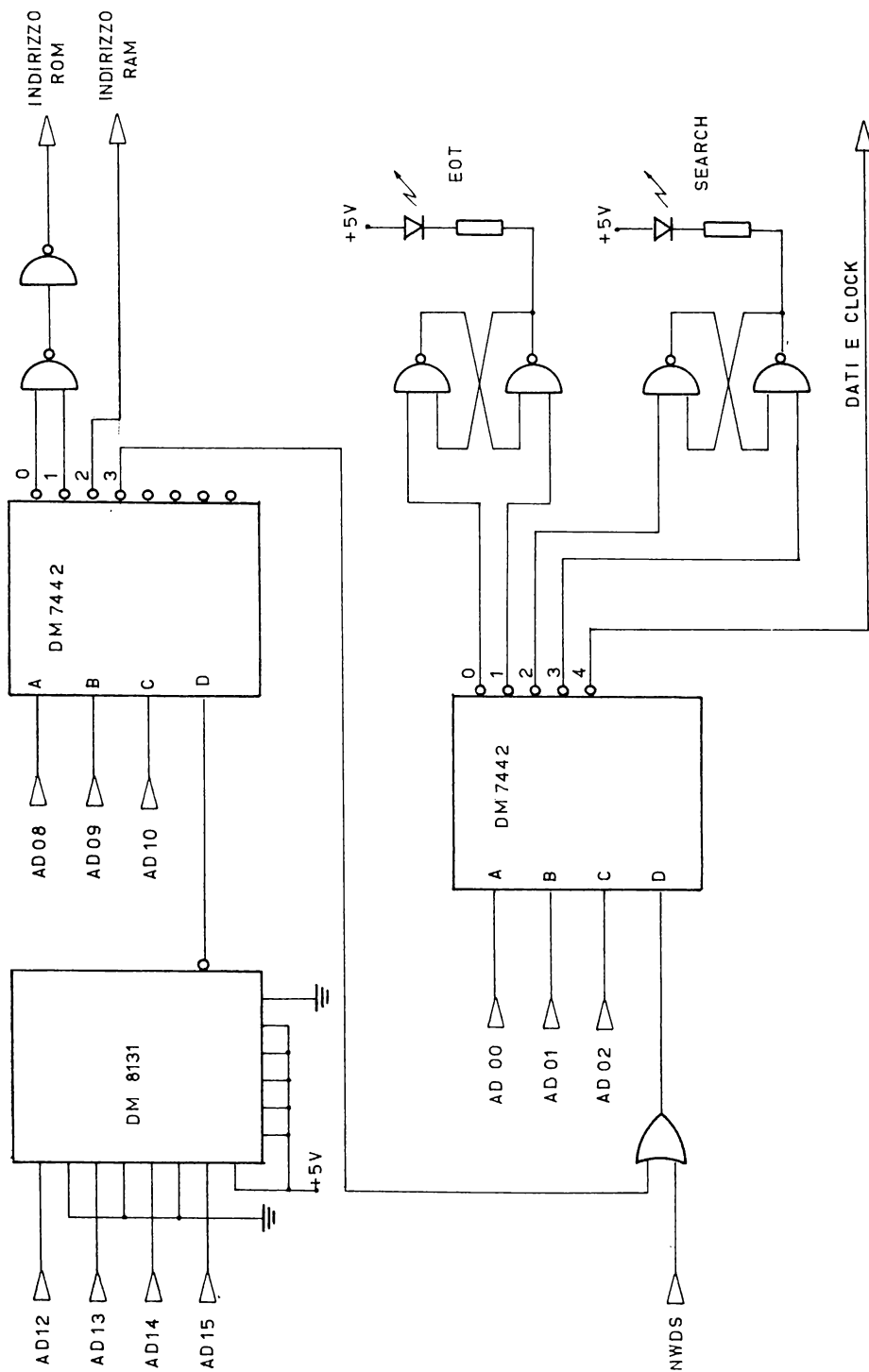
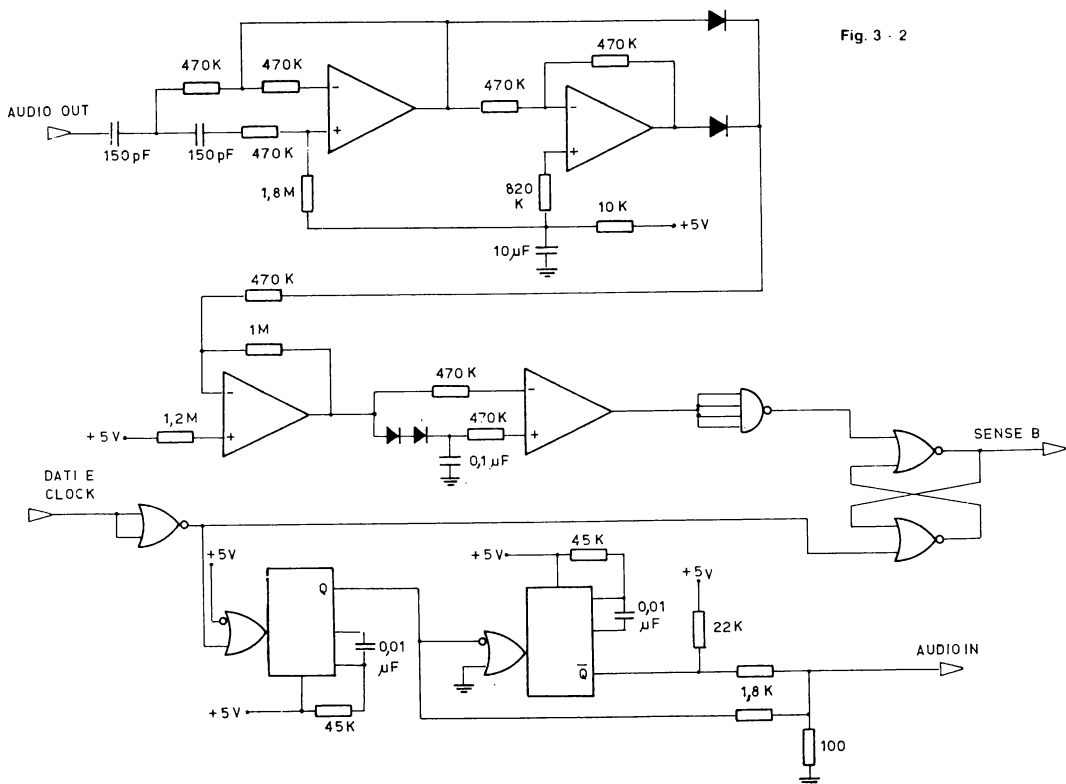
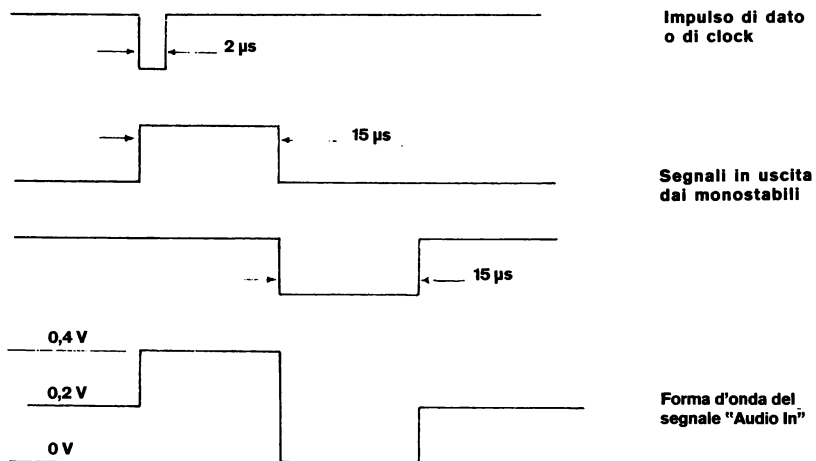
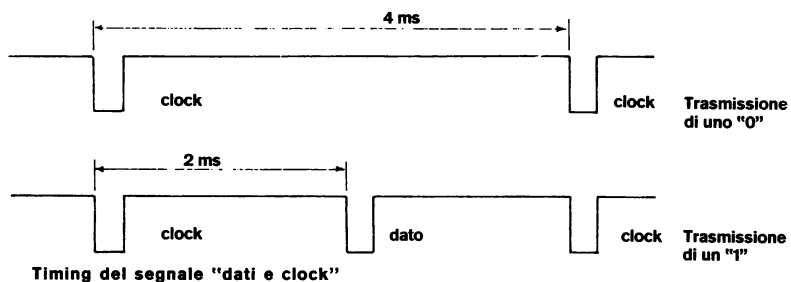
segue

Infine l'informazione relativa alla lunghezza del blocco viene caricata nelle locazioni X'820A ed X'820B. L'operatore metterà in funzione il registratore e quindi farà eseguire al processore il programma di scrittura, che inizia dalla locazione X'80C7. L'indicatore di "Search" si accende dopo che il processore ha scritto il blocco di leader e si spegne quando la trasmissione è terminata. A questo punto, viene acceso l'indicatore di "End of Trasmission", mentre il processore andrà in halt all'indirizzo X'8142. Per leggere la cassetta magnetica, l'utente dovrà far eseguire a SC/MP la routine di loader, che inizia dall'indirizzo X'8000. Deve quindi essere acceso il registratore, che deve essere posto in lettura. L'indicatore di "End of Trasmission" viene acceso quando la lettura di un blocco è terminata, mentre l'indicatore di "Search" resta acceso sinché non viene trovata la parola d'identificazione, quindi viene spento. Se la lettura del blocco è stata eseguita in modo corretto, l'indicatore di "Search" viene poi acceso di nuovo a fine lettura. Durante le operazioni di lettura il controllo del volume del registratore deve essere tenuto in posizione tale che il segnale in uscita dallo stesso sia al limite della saturazione; è importante notare, in ogni caso, che il volume può essere regolato semplicemente per tentativi (qualora non sia possibile osservare il segnale) sinché non si ottiene una lettura esente da errori.

Fig. 3 - 2

Fig. 3 - 2

Timing del segnale "dati e clock"

Diagrammi dei tempi

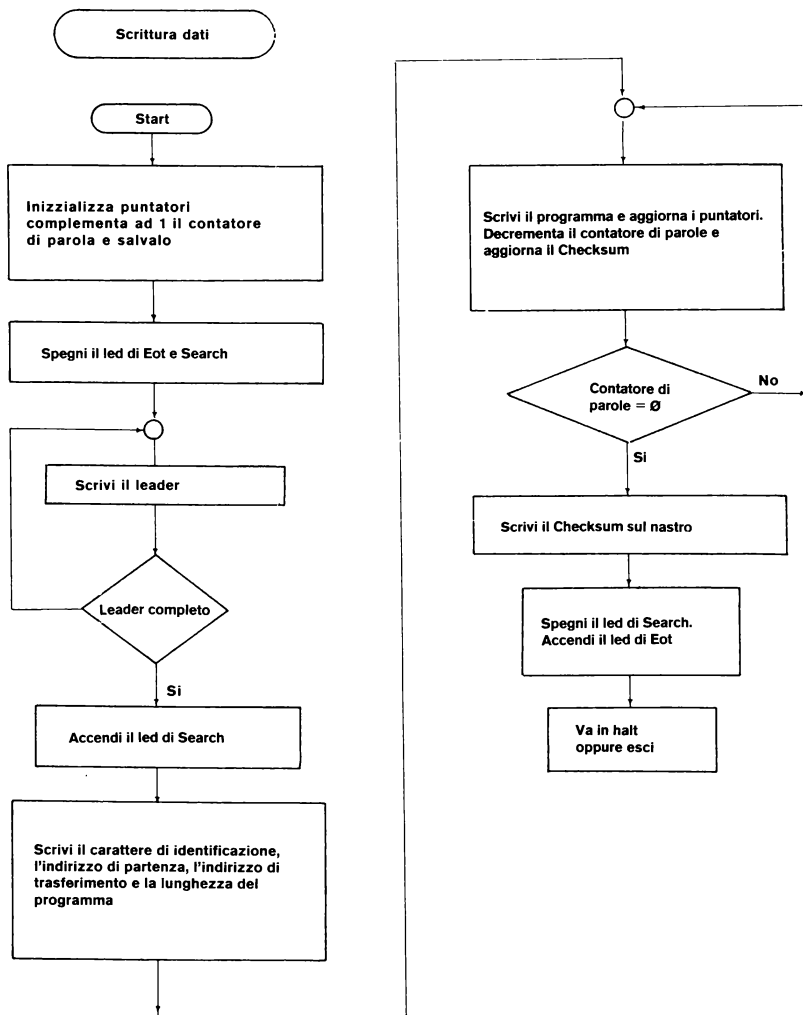



Fig. 3 - 3. Flowchart del programma di scrittura e lettura in cassetta audio (Continua)

segue

Lettura dati
Start
Inizializza i puntatori e le celle di salvataggio. Accendi il led di Search e spegni quello di Eot
Ricerca del carattere di identificazione
Ricevuto
No
Si
Spegni il led di Search
Metti in P3 l'indirizzo di partenza letto mentre in RAM quello di trasferimento e il contatore di parole lette
Leggi dal nastro i bytes del programma e memorizzalo in RAM indirizzata da un puntatore di indice. Aggiorna Checksum e decrementa il contatore di parole
Si
No
Contatore di parole = Ø
Leggi dal nastro il Checksum e paragonalo con quello calcolato
No
Uguale
Si
Accendi il led di Eot e di Search per indicare caricamento avvenuto. Salta al programma utente
Accendi il led di Eot. Halt per errore
Esci

Fig. 3-4. Listing programma di scrittura e lettura in cassetta audio

```
NEXT ASSEMBLY
*.ASM PT,OM

END PASS 1
     1                     .TITLE TAPEIO, 'SC/MP ROUTINES'
     2            ;
     3    8000             .=X'8000
     4            ;
     5    8200             PAM     =X'8200      ; PUNTATORE RAM
     6    8300             PERIPH  =X'8300      ; INDIR. PERIF.
     7            ;
     8    0003             P3      =3           ; PUNTAT. 3
     9    0002             P2      =2           ; PUNTAT. 2
    10    0001             P1      =1           ; PUNTAT. 1
    11            ;
    12                     ; CELLE RAM PER MEMORIZZAZIONI
    13                     ; PROVVISORIE DI DATI.
    14            ;
    15    0000             CNTU    =0           ; CONTAT. INTERNO
    16                                          ; RELATIVO AL LEADER
    17    0001             CNTL    =1           ; CONTAT. ESTERNO
    18                                          ; RELATIVO AL LEADER
    19    0002             CKSUM   =2           ; CONTAT. DI CHECKSUM
    20    0003             STARTU  =3           ; IND. START (BY. PES.)
    21    0004             STARTL  =4           ; IND. START (BY. LEG.)
    22    0005             BITCNT  =5           ; CONTATORE BIT
    23    0006             TEMP1   =6           ; CELLA DI SALV.
    24    0007             TEMP2   =7           ; CELLA DI SALV.
    25    0008             TEMP3   =8           ; CELLA DI SALV.
    26    0009             TEMP4   =9           ; CELLA DI SALV.
    27    000A             WDCNTU  =10          ; CONT. PAROLE (BY. PES.)
    28    000B             WDCNTL  =11          ; CONT. PAROLE (BY. LEG.)
ERROR DUP. DEF.                     ?
    29    000C             JUMPU   =12          ; IND. TRASFER. (BY. PES.)
    30    000D             JUMPL   =13          ; IND. TRASFER. (BY. LEG.)
    31            ;
    32                     ; CODICI ORDINI PERIFERICA
    33            ;
    34    0000             EOTON   =0           ; LED EOT IN ON
    35    0001             EOTOFF  =1           ; LED EOT IN OFF
    36    0002             SRCHON  =2           ; LED SEARCH IN ON
    37    0003             SRCHOF  =3           ; LED SEARCH IN OFF
    38    0004             FLAG    =4           ; FLAG DI READ/WRITE
    39            ;
    40            ;
    41


TAPEIOSC/MP ROUTINES
BOOTSTRAP LOADER

                           .PAGE 'BOOTSTRAP LOADER'
    42            ;
    43                     ; ROUTINE DI BOOTSTRAP. CARICA UN
    44                     ; PROGRAMMA MESSO SU NASTRO MAGNETICO.
    45                     ; TUTTE LE INFORMAZIONI NECESSARIE
    46                     ; AL CARICAMENTO SONO SUL NASTRO.
    47            ;
    48                     ; QUESTO PROGRAMMA PUÒ ESSERE RIAS-
    49                     ; SEMBLATO ALL'INDIRIZZO X'0000 IN
    50                     ; MODO DA POTER ESEGUIRE IL CARICAMENTO
    51                     ; ALL'ACCENSIONE.
    52            ;
    53                     ; SIGNIFICATO DEGLI INDICATORI A LED
    54                     ; DURANTE LE OPERAZIONI DI INPUT:
    55            ;
    56                     ; LED SEARCH IN ON ALLA PARTENZA DEL PROGR.
```

segue

Fig. 3 - 4

```
 57                           ; LED SEARCH IN OFF ALLA RICEZIONE DEL
 58                           ; CARATTERE DI IDENTIFICAZIONE
 59                           ; LED EOT IN ON A CARICAM. COMPLETATO
 60                           ; LED SEARCH IN ON SE C'È CHECKSUM
 61                   ;
 62                           ; IL CONTROLLO È POI TRASFERITO AL
 63                           ; PROGRAMMA UTENTE.
 64                   ;
 65                   ;
 66  8000  08    BOOT:    NOP                      ; PER RILOCARE IL PROGR.
 67  8001  C400           LDI     L (RAM)          ; A X'0000 INIZIAL. PUNT.
 68  8003  32             XPAL    P2               ; RAM IN P2
 69  8004  C482           LDI     H (RAM)
 70  8006  36             XPAH    P2
 71  8007  C400           LDI     0                ; AZZERA AC
 72  8009  CA02           ST      CKSUM (P2)       ; INIZ. CONTAT. CHECKSUM
 73  800B  C400           LDI     L (PERIPH)       ; IN P3 IND. PERIFERICA
 74  800D  33             XPAL    P3
 75  800E  C483           LDI     H (PERIPH)
 76  8010  37             XPAH    P3
 77  8011  CB02           ST      SRCHON (P3)      ; IN ON LED SEARCH
 78  8013  CB01           ST      EOTOFF (P3)      ; IN OFF LED EOT
 79  8015  C400           LDI     0                ; AZZERA AC
 80  8017  01             XAE                      ; AZZERA REG. E
 81  8018  C48F           LDI     L (GETBIT)-1     ; IN P1 IND. ROUTINE
 82  801A  31             XPAL    P1               ; GETBIT
 83  801B  C480           LDI     H (GETBIT)
 84  801D  35             XPAH    P1
 85  801E  3D    LOCID:   XPPC    P1               ; VA ALLA ROUT. GETBIT
 86  801F  40             LDE
 87  8020  E4A5           XRI     X'A5             ; VERIF. CAR. IDENTIF.
 88  8022  9802           JZ      SETPNT           ; SE RICEVUTO SALTA
 89  8024  90F8           JMP     LOCID            ; SE NO PRENDI BIT SUCC.
 90  8026  CB03  SETPNT:  ST      SRCHOF (P3)      ; LED SEARCH IN OFF
 91  8028  C46D           LDI     L (RECV)-1       ; IN P1 IND. ROUTINE
 92  802A  31             XPAL    P1               ; RECV
 93  802B  C480           LDI     H (RECV)
 94  802D  35             XPAH    P1
 95  802E  3D             XPPC    P1               ; VA ALLA ROUT. RECV
 96  802F  33             XPAL    P3
 97  8030  3D             XXPC    P1
 98  8031  37             XPAH    P3
 99  8032  3D             XPPC    P1               ; RICAVA IND. TRASFER.
100  8033  CA0D           ST      JUMPL (P2)       ; E SALVALO IN RAM
101  8035  3D             XPPC    P1
102  8036  CA0C           ST      JUMPU (P2)
103  8038  3D             XPPC    P1               ; RICAVA CONTAT. PAROLE
104  8039  CA0B           ST      WDCNTL (P2)      ; E SALVALO IN RAM
105  803B  3D             XPPC    P1
106  803C  CA0A           ST      WDCNTU (P2)
107                   ;
108                   ;
109  803E  3D    BOOTIN:  XPPC    P1               ; VA ALLA ROUT. RECV
110  803F  CF01           ST      @1 (P3)          ; MEMOR. ED INCR. PUNT.
111  8041  F202           ADD     CKSUM (P2)       ; AGGIORNA CHECKSUM
112  8043  CA02           ST      CKSUM (P2)
113  8045  CA0B           ILD     WDCNTL (P2)      ; INCR. BY. LEG. CON. PAR.
114  8047  9CF5           JNZ     BOOTIN           ; VERIFICA SE ZERO
115  8049  AA0A           ILD     WDCNTU (P2)      ; INCR. BY. PES. CON. PAR.
116  804B  9CF1           JNZ     BOOTIN           ; VERIF. SE È EOT
117  804D  3D             XPPC    P1               ; LEGGI CHECK. DAL NASTRO
118  804E  E202           XOR     CKSUM (P2)       ; VERIF. SE = VAL. CALC.
119  8050  9809           JZ      EXECPR           ; ESEGUI PROGR. CARICATO
120  8052  C400           LDI     L (PERIPH)
121  8054  33             XPAL    P3
122  8055  C483           LDI     H (PERIPH)
123  8057  37             XPAH    P3
124  8058  CB00           ST      EOTON (P3)       ; LED EOT IN ON PER
125  805A  00             HALT                     ; INDIC. ERR. DI CHECKSUM
```

segue

Fig. 3 - 4

```
126                     ;
127  805B  C400  EXECPR:  LDI    L (PERIPH)
128  805D  33             XPAL   P3
129  850E  C483           LDI    H (PERIPH)
130  8060  37             XPAH   P3
131  8061  CB00           ST     EOTON (P3)     ; LED EOT IN ON
132  8063  CB02           ST     SRCHON (P3)    ; LED SEARCH IN ON
133  8065  C20D           LD     JUMPL (P2)     ; IN P3 IND. DI
134  8067  33             XPAL   P3             ; TRASFERIMENTO
135  8068  C20C           LD     JUMPU (P2)
136  806A  37             XPAH   P3
137  806B  C7FF           LD     @ -1 (P3)      ; DECREM. PUNTAT. PER
138                                             ; INIZ. FASE FETCH
139  806D  3F             XPPC   P3             ; ESECUZ. PROGRAMMA
140                     ;
141                     ;
142                          ; ROUTINE DI RICEZIONE. PRENDE DAL
143                          ; NASTRO UN CARATTERE DI 8 BIT E LO
144                          ; DEPOSITA IN AC.
145                     ;
146  806E  C48F  RECV:    LDI    L (GETBIT)-1   ; IN P1 INDIRIZZO
147  8070  31             XPAL   P1             ; ROUT. GETBIT
148  8071  CA07           ST     TEMP2 (P2)     ; MEMOR. P1 CORRENTE
149  8073  C480           LDI    H (GETBIT)
150  8075  35             XPAH   P1
151  8076  CA06           ST     TEMP1 (P2)
152  8078  C408           LDI    8              ; INIZIAL. CONTAT. BIT
153  807A  CA05           ST     BITCNT (P2)
154  807C  C400           LDI    0              ; AZZERA AC
155  807E  01             XAE                   ; AZZERA REG. E
156  807F  3D    LOOP:    XPPC   P1             ; VA ALLA ROUT. GETBIT
157  8080  BA05           DLD    BITCNT (P2)    ; DECREM. CONTAT. BIT
158  8082  9802           JZ     RETRN2         ; VERIFICA SE È ZERO
159  8084  90F9           JMP    LOOP
160  8086  C207  RETRN2:  LD     TEMP2(P2)      ; RIPRISTINA IN P1
161  8088  31             XPAL   P1             ; VALORE ORIGINALE
162  8089  C206           LD     TEMP1(P2)
163  808B  35             XPAH   P1
164  808C  40             LDE                   ; IN AC CARATTERE
165  808D  3D             XPPC   P1             ; RITORNO DALLA ROUT.
166  808E  90DE           JMP    RECV
167                     ;
168                     ;
169                          ; ROUTINE DI LETTURA BIT. IL BIT LETTO
170                          ; È MESSO NEL REGISTRO E
171                     ;
172  8090  C400  GETBIT:  LDI    L(PERIPH)      ; IN P3 INDIRIZZO
173  8092  33             XPAL   P3             ; PERIFERICA
174  8093  CA09           ST     TEMP 4 (P2)    ; MEMOR. P3 CORRENTE
175  8095  C483           LDI    H(PERIPH)
176  8097  37             XPAH   P3
177  8098  CA08           ST     TEMP3(P2)
178  809A  19             SIO                   ; SHIFTA REG. E
179  809B  06    CKSA:    CSA                   ; STATUS IN AC
180  809C  D420           ANI    X'20           ; MASCHERA
181  809E  9802           JZ     CLOCK          ; SE STATUS = 0, BIT RIC.
182  80A0  90F9           JMP    CKSA           ; RIPROVA
183  80A2  C400  CLOCK:   LDI    0              ; AZZ. AC PER FARE RITARDO
184  80A4  8F01           DLY    1              ; RITARDO = 1MS (1/4 BIT)
185  80A6  CB04           ST     FLAG(P3)       ; RESET LATCH
186  80A8  C400           LDI    0              ; AZZ. AC PER FARE RITARDO
187  80AA  8F02           DLY    2
188  80AC  06             CSA                   ; STATUS IN AC
189  80AD  D420           ANI    X'20           ; MASCHERA STATUS
190  80AF  9802           JZ     ONE            ; SE STATUS = 0, BIT = 1
191  80B1  9004           JMP    RESET
192  80B3  40    ONE:     LDE
193  80B4  DC80           ORI    X'80           ; SOMMA BIT "1" AL CAR.
194  80B6  01             XAE                   ; CARAT. IN REG. E
```

segue

Fig. 3 - 4

```
195  80B7  CB04  RESET:   ST    FLAG(P3)      ; RESET LATCH
196  80B9  06              CSA                 ; STATUS IN AC
197  80BA  D420            ANI   X'20          ; MASCHERA STATUS
198  80BC  98F9            JZ    RESET         ; VERIF. SE LATCH È RESET.
199  80BE  C209  RETRN3:   LD    TEMP4(P2)     ; RIPRISTINA IN P3
200  80C0  33              XPAL  P3            ; VALORE ORIGINALE
201  80C1  C208            LD    TEMP3(P2)
202  80C3  37              XPAH  P3
203  80C4  3D              XPPC  P1            ; RITORNO DALLA ROUT.
204  80C5  90C9            JMP   GETBIT
205                  ;
206                  ;
207
```

TAPEIOSC/MP ROUTINES
DATA WRITE ROUTINES

```
                           .PAGE 'DATA WRITE ROUTINES'
208                  ;
209                        ; INVIA PER 4 SECONDI IL CARATTERE
210                        ; "0" PER PERMETTERE AL NASTRO DI
211                        ; POSIZIONARSI SUL PLAY-BLACK E PER
212                        ; RICAVARE IL LEADER.
213                  ;
214                        ; SIGNIFICATO DEGLI INDICATORI A LED
215                        ; DURANTE LE OPERAZIONI DI OUTPUT:
216                  ;
217                        ; LED SEARCH IN ON A COMPLETAMENTO
218                        ; DEL LEADER
219                        ; LED SEARCH IN OFF A FINE TRASMISSIONE
220                        ; LED EOT IN ON A FINE TRASMISSIONE
221                  ;
222                  ;
223  80C7  C400  INIT:     LDI   L(RAM)        ; IN P2 PUNTATORE
224  80C9  32              XPAL  P2            ; RAM
225  80CA  C482            LDI   H (RAM)
226  80CC  36              XPAH  P2
227  80CD  C400            LDI   L(PERIPH)     ; IN P3 INDIRIZZO
228  80CF  33              XPAL  P3            ; PERIFERICA
229  80D0  C483            LDI   H(PERIPH)
230  80D2  37              XPAH  P3
231  80D3  C400  COMP:     LDI   0             ; AZZERA AC
232  80D5  02              CCL                 ; AZZ. FLAG CARRY/LINK
233  80D6  FA0B            CAD   WDCNTL(P2)    ; COMPLEMENTA AD 1
234  80D8  CA0B            ST    WDCNTL(P2)    ; BYTE LEGGERO CONT.
235  80DA  C400            LDI   0
236  80DC  FA0A            CAD   WDCNTU(P2)    ; COMPLEMENTA AD 1
237  80DE  CA0A            ST    WDCNTU(P2)    ; BYTE PESANTE CONT.
238  80E0  CB03            ST    SRCROF(P3)    ; LED SEARCH IN OFF
239  80E2  CB01            ST    EOTOFF(P3)    ; LED EOT IN OFF
240  80E4  C408  SNDLDR:   LDI   8             ; INIZIAL. CONTAT.
241  80E6  CA01            ST    CNTL(P2)      ; ESTERNO
242  80E8  C480  CNT1:     LDI   X'80          ; INIZIAL. CONTAT.
243  80EA  CA00            ST    CNTU(P2)      ; INTERNO
244  80EC  CB04  CNT2:     ST    FLAG(P3)      ; PULSA FLAG DI WRITE
245  80EE  C400            LDI   0             ; AZZERA AC
246  80F0  8F04            DLY   4             ; RITARDO DI 1 BIT
247  80F2  BA00            DLD   CNTU(P2)      ; DECR. CONTAT. INTERNO
248  80F4  9CF6            JNZ   CNT2          ; VERIFICA SE = 0
249  80F6                  DLD   CNTL(P2)      ; DECR. CONTAT. ESTERNO
250  80F8  94EE            JP    ONT1          ; VERIFICA SE < 0
251  80FA  CB02            ST    SRCHON (P3)   ; LED SEARCH IN ON
252                  ;
253                  ;
254                        ; ROUTINE DI TRASFERIMENTO BLOCCHI.
255                        ; INVIA IL BLOCCO DI DATI ALLA CASSETTA.
256                  ;
257                        ; PRIMA DI ESEGUIRE LA ROUTINE WRITE
258                        ; L'UTENTE DEVE CARICARE I SEGUENTI
```

segue

Fig. 3 - 4

```
259                               ; INDIRIZZI:
260                     ;
261                               ; X'8203 IND. PROGR. DA CARICARE (BY. PES.)
262                               ; X'8204 IND. PROGR. DA CARICARE (BY. LEG.)
263                               ; X'820A LUNGHEZZA PROGRAMMA (BYTE PESANTE)
264                               ; X'820B LUNGHEZZA PROGRAMMA (BYTE LEGGERO)
265                               ; X'820C IND. TRASFERIMENTO (BYTE PESANTE)
266                               ; X'820D IND. TRASFERIMENTO (BYTE LEGGERO)
267                     ;
268                     ;
269  80FC  C400  BLOCK:   LDI    0             ; AZZERA AC
270  80FE  CA02           ST     CKSUM(P2)     ; INIZ. CONT. CHECKSUM
271  8100  C481           LDI    H(WRITE)      ; IN P1 IND. ROUTINE
272  8102  35             XPAH   P1            ; WRITE
273  8103  C444           LDI    L(WRITE)-1
274  8105  31             XPAL   P1
275  8106  C4A5           LDI    X'A5          ; IN AC CARAT. IDENTIF.
276  8108  3D             XPPC   P1            ; SCRIVILO SUL NASTRO
277  8109  C204           LD     STARTL(P2)    ; IN AC IND. START
278  810B  3D             XPPC   P1            ; SCRIVILO SUL NASTRO
279  810C  C203           LD     STARTU(P2)
280  810E  3D             XPPC   P1
281  810F  C20D           LD     JUMPL(P2)     ; IN AC IND. TRASFERIM.
282  8111  3D             XPPC   P1            ; SCRIVILO SUL NASTRO
283  8112  C20C           LD     JUMPU(P2)
284  8114  3D             XPPC   P1
285  8115  C20B           LD     WDCNTL(P2)    ; IN AC LUNGHEZZA
286  8117  3D             XPPC   P1            ; SCRIVILA SUL NASTRO
287  8118  C20A           LD     WDCNTU(P2)
288  811A  3D             XPPC   P1
289  811B  C204  GETBYT:  LD     STARTL(P2)    ; IN P1 IND. CORRENTE
290  811D  31             XPAL   P1
291  811E  C203           LD     STARTU(P2)
292  8120  35             XPAH   P1
293  8121  C501           LD     @1(P1)        ; CARATT. IN ACC.
294  8123  01             XAE
295  8124  C444           LDI    L(WRITE)-1    ; IN P1 IND. ROUTINE
296  8126  31             XPAL   P1            ; WRITE CON MEMORIZ.
297  8127  CA04           ST     STARTL(P2)    ; VALORE CORRENTE
298  8129  C481           LDI    H(WRITE)
299  812B  25             XPAH   P1
300  812C  CA03           ST     STARTU(P2)
301  812E  40             LDE
302  812F  F202           ADD    CKSUM(P2)     ; AGGIORNA CHECKSUM
303  8131  CA02           ST     CKSUM(P2)
304  8133  40             LDE                  ; IN AC CARATTERE
305  8134  3D             XPPC   P1            ; SCRIVILO SUL NASTRO
306  8135  AA0B           ILD    WDCNTL(P2)    ; INCR. CONT. PAROLE
307  8137  9CE2           JNZ    GETBYT        ; VERIFICA SE = 0
308  8139  AA0A           ILD    WDCNTU(P2)
309  813B  9CDE           JNZ    GETBYT
310  813D  C202           LD.    CKSUM(P2)     ; SCRIVI CHECKSUM
311  813F  3D             XPPC   P1            ; SUL NASTRO
312  8140  CB03           ST     SRCHOF(P3)    ; LED SEARCH IN OFF
313  8142  CB00           ST     EOTON(P3)     ; LED EOT IN ON
314  8144  00             HALT
315                     ;
316                     ;
317                               ; ROUTINE DI SCRITTURA DEI DATI.
318                               ; SCRIVE UN CARATTERE A 8 BIT SUL NASTRO
319                     ;
320  8145  01    WRITE:   XAE                  ; IN REG. E CARAT.
321  8146  C408           LDI    8             ; INIZIAL. CONT. BIT
322  8148  CA05           ST     BITCNT(P2)
323  814A  40    MASK:    LDE
324  814B  D401           ANI    1             ; MASCHERA CARAT.
325  814D  9C08           JNZ    SENDI         ; VERIF. SE BIT 0 OP 1
326  814F  C400           LDI    0             ; AZZERA AC
327  8151  CB04  SEND0:   ST     FLAG(P3)      ; PULSA FLAG WRITE
```

segue

Fig. 3 - 4

```
321  8146  C408            LDI    8            ; INIZIAL. CONT. BIT
322  8148  CA05            ST     BITCNT(P2)
323  814A  40     MASK:    LDE
324  814B  D401            ANI    1            ; MASCHERA CARAT.
325  814D  9C08            JNZ    SEND1        ; VERIF. SE BIT 0 OP 1
326  814F  C400            LDI    0            ; AZZERA AC
327  8151  CB04   SEND1:   ST     FLAG(P3)     ; PULSA FLAG WRITE
328  8153  8F04            DLY    4            ; RITARDO DI 1 BIT
329  8155  900C            JMP    SHIFT
330  8157  C400   SEND1:   LDI    0
331  8159  CB04            ST     FLAG(P3)     ; PULSA FLAG WRITE
332  815B  8F02            DLY    2
333  815D  CB04            ST     FLAG(P3)     ; PULSA FLAG WRITE
334  815F  C400            LDI    0
335  8161  8F02            DLY    2
336                ;
337  8163  19     SHIFT:   SIO                 ; SHIFTA REG. E
338  8164  BA05            DLD    BITCNT(P2)   ; DECR. CONT. BIT
339  8166  9802            JZ     RETRN1       ; VERIFICA SE = 0
340  8168  90E0            JMP    MASK         ; INVIA BIT SUCCES.
341  816A  3D     RETRN1:  XPPC   P1           ; RITORNO DALLA ROUT.
342  816B  90D8            JMP    WRITE
343                ;
344        8000            .END BOOT

BITCNT   0005      BLOCK    80FC*     BOOT     8000
BOOTIN   803E      CKSA     809B      CKSUM    0002
CLOCK    80A2      CNT1     80E8      CNT2     80EC
CNTL     0001      CNTU     0000      COMP     80D3*
EOTOFF   0001      EOTON    0000      EXECPR   805B
FLAG     0004      GETBIT   8090      GETBYT   811B
INIT     80C7*     JUMPL    000D      JUMPU    000C
LOCID    801E      LOOP     807F      MAS'     814A
ONE      80B3      P1       0001      P2       0002
P3       0003      PERIPH   8300      RAM      8200
RECV     806E      RESET    80B7      RETRN1   816A
RETRN2   8086      RETRN3   80BE*     SEND0    8151*
SEND1    8157      SETPNT   8026      SHIFT    8163
SNDLDR   80E4*     SRCHOF   0003      SRCHON   0002
STARTL   0004      STARTU   0003      TEMP1    0006
TEMP2    0007      TEMP3    0008      TEMP4    0009
WDCNTL   000B      WDCNTU   000A      WRITE    8145

NO ERROR LINES
END PASS 2
SOURCE CHECKSUM = 2DD6
```

Capitolo 4°

SISTEMA DI INTERRUZIONE DEL MICROPROCESSORE SC/MP

4.1 Interrupt a singolo livello

La logica di interruzione dello SC/MP è indicata nella fig. 4-1. Prima del fetch di un'istruzione, il bit 3 dello status register è testato. Se il bit è nello stato logico 0 (flag di interrupt enable non settato) e l'input BB CONT è nello stato logico 1, il program counter è incrementato e si attua il fetch e l'esecuzione della successiva istruzione. Se invece l'interrupt enable è allo stato logico 1 ed il Sense A è alto, viene servita l'interruzione. Come conseguenza l'interrupt enable è resettato ed il contenuto del program counter è scambiato con il contenuto del pointer 3: questo registro puntatore contiene l'indirizzo della routine che gestisce l'interruzione. Quindi il pointer 3 deve essere inizializzato con l'indirizzo della routine di servizio dell'interrupt; successivamente occorre abilitare il sistema di interruzione ponendo ad 1 il flag di Interrupt Enable. Ad esempio, volendo abilitare il sistema di interruzione in un dato punto del programma principale ed avendo la routine di servizio dello interrupt alla locazione di memoria X'1020, occorre implementare la seguente serie di istruzioni:

```
Programma principale
.
.
.
LDI    X'20  ; Carica l'accumulatore con il byte di or-
             ; dine più basso dell'indirizzo della rou-
             ; tine di interruzione (X'20)
XPAL   P3    ; Poni il byte di ordine più basso in P3L
LDI    X'10  ; Carica l'accumulatore con il byte di or-
             ; dine più alto dell'indirizzo della routine
             ; di interruzione (X'10)
XPAH   P3    ; Poni il byte di ordine più alto in P3H
IEN          ; Abilitazione dell'interrupt (set del bit 3
             ; dello status register)
.
Prosecuzione del programma principale
```

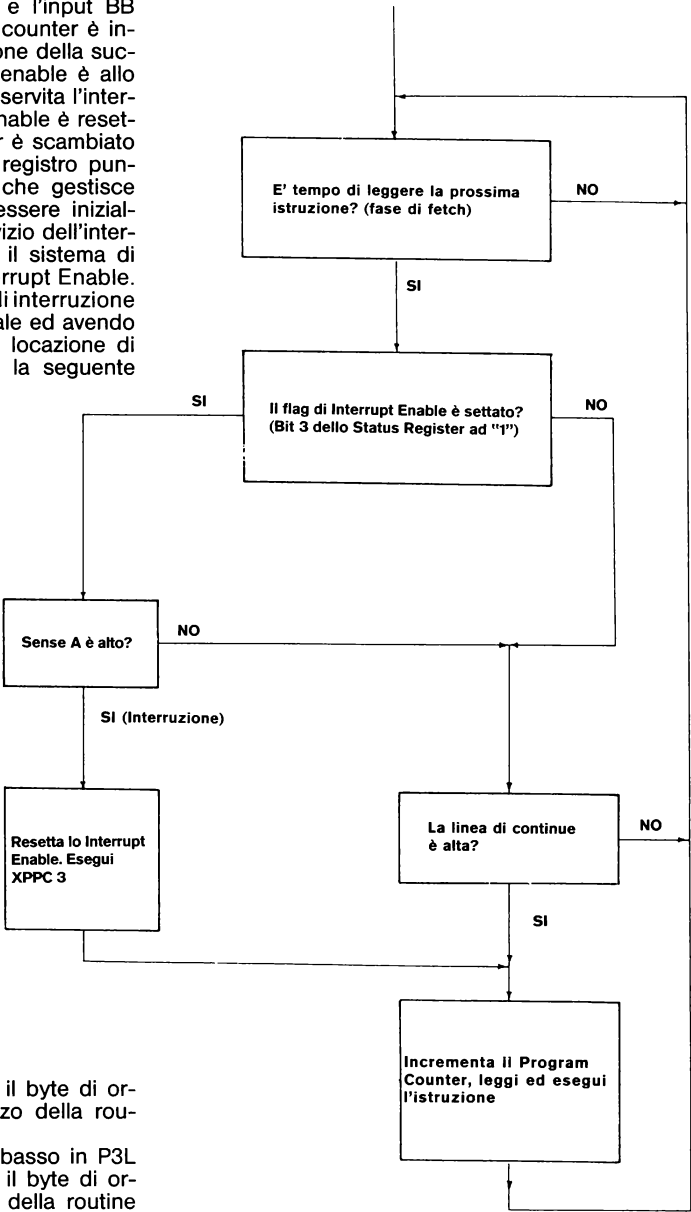



Fig. 4 - 1. Flow delle operazioni di Fetch/Interrupt in SC/MP

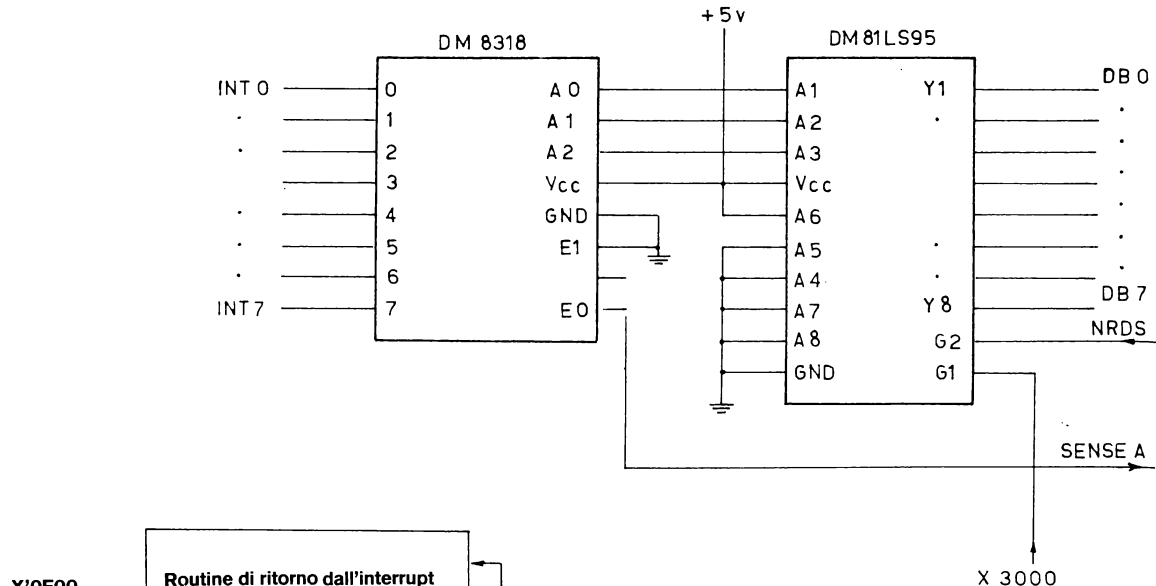



Fig. 4 - 2. Utilizzo di un priority encoder per realizzare un sistema di interrupt a più livelli (a) e organizzazione in memoria delle routine di gestione dell'interruzione (b).

Fig. 4 - 2 (b)

4.2 Interrupt Multi-Level

Un sistema con più di un livello di interruzione può essere facilmente implementato interfacciando un priority encoder allo SC/MP. In questo sistema, la routine di servizio dell'interrupt per ogni dispositivo richiedente interruzione deve essere attentamente scritta per evitare complicazioni software nell'ambito della gestione delle varie interruzioni. Questa considerazione è molto valida in riferimento ai salvataggi dello stato attuale della macchina, al verificarsi della interruzione. Il salvataggio di stato può facilmente essere complicato, specialmente in un sistema con interruzioni a priorità nidificate; in questo caso il livello a priorità superiore può interrompere la routine di servizio dell'interrupt a priorità inferiore. Dato che le routine relative agli interrupt nidificati sono funzione del software dell'utilizzatore, non sono qui trattate. La fig. 4-2 mostra un metodo di interfacciamento dello SC/MP con un encoder di priorità. Ciascuno degli otto input (da INT 0 a INT 7) proviene da un output di un flip-flop resettabile da SC/MP durante l'esecuzione della routine di servizio della interruzione. L'encoder pone in OR gli input di interrupt; se una qualunque delle linee diventa attiva ("active low"), la linea Sense A del microprocessore viene posta alta tramite la linea di output E0. Se uno o più input richiedono servizio di interrupt nello stesso istante, interviene il sistema di priorità: INT 7 è l'interrupt con più elevata priorità, INT 0 è l'interrupt a più bassa priorità. Gli output A0, A1, A2 dell'encoder costituiscono un codice binario a 3 bit che corrisponde all'interrupt di livello più elevato; questo codice forma la parte meno significativa della "interrupt data word", il cui ingresso allo

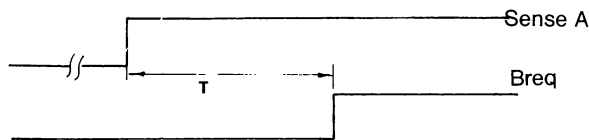
SC/MP avviene tramite un buffer TRI-STATE. L'output del buffer fornisce gli altri 8 bit dell'indirizzo a 16 bit richiesto per ogni routine di servizio di interruzione. Come mostrato in fig. 4-2, l'interrupt a priorità più elevata (INT 7), è assegnato in modo arbitrario all'indirizzo X'2000, il successivo livello corrisponde a X'2100 ed il successivo ancora ha come indirizzo X'2200 e così via per i rimanenti livelli. Di seguito sono indicate le "output data word" per ogni livello.

Livello di interruzione	Interrupt buffer output Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
7	0	0	1	0	0	0	0	0
6	0	0	1	0	0	0	0	1
5	0	0	1	0	0	0	1	0
4	0	0	1	0	0	0	1	1
3	0	0	1	0	0	1	0	0
2	0	0	1	0	0	1	0	1
1	0	0	1	0	0	1	1	0
0	0	0	1	0	0	1	1	1

Le allocazioni in memoria delle otto routine di servizio delle interruzioni (INT 7 ed INT 0) sono indicate nella fig. 4-2. Queste allocazioni includono la routine di start dell'interrupt, la routine di ritorno dall'interrupt, la routine di memorizzazione temporanea dello stato della macchina. Gli indirizzi sono scelti in modo tale da essere compresi nella medesima pagina, permettendo in questo modo l'utilizzo dell'indirizzamento relativo al program counter per la routine start/return. La routine di start dell'interrupt salva lo stato attuale della macchina (Accumulatore, Program Counter, Pointer, Register e così via) e dirige il programma verso la routine di servizio interruzione dell'interrupt a priorità più elevata. Per trasferire il controllo alla routine di servizio, il programma azzera la metà più bassa del pointer (pointer P2, nell'esempio). Successivamente il programma legge l'address data word (0010 0XXX) dell'interrupt a priorità più alta; questa informazione è trasferita alla metà superiore del pointer, che fornisce l'indirizzo a 16 bit della richiesta routine di servizio. A questo punto il program counter ed il pointer sono scambiati ed è posta in esecuzione la routine di servizio dell'interrupt. Come mostrato in fig. 4-2, ogni interrupt deve passare per un routine di start. Occorre in ogni caso notare che non appena un interrupt è riconosciuto, tutti gli interrupt in atto, anche quelli a priorità più elevata, devono attendere il completamento dell'interruzione in atto prima di essere serviti. Le routine di servizio interrupt da 0 a 7 contengono il programma per servire il particolare dispositivo che richiede l'interruzione. Tipicamente questi programmi attuano un trasferimento input/output di dati, esercitano un controllo meccanico/elettrico di periferiche, forniscono timing per sincronizzazioni ed altre funzioni. Il flow-chart ed il listing del programma fornito, sono da considerarsi come uno schema generale flessibile che rappresenta uno dei molti modi per implementare un sistema di interruzione a multi-level. Ad esempio, le routine di servizio possono partire da locazioni diverse da X'2000 semplicemente modificando gli input al buffer TRI-STATE nella fig. 4-2.

Ancora, le routine non devono essere necessariamente lunghe 256 byte; il numero delle parole può essere alterato modificando gli input al buffer e caricando la metà bassa del pointer invece della metà alta. Prima del completamento della routine di servizio, tutti gli interrupt devono passare attraverso una routine di ritorno dall'interrupt. Questa routine essenzialmente ripristina lo stato della macchina; l'interrupt enable è settato per mettere il microprocessore in condizione di servire gli altri interrupt eventualmente in attesa. Le routine di start e di ritorno possono essere modificate in funzione del numero di registri da salvare. Con riferimento al pointer P3, occorre notare che è utilizzato in un unico modo; in particolare P3 è utilizzato come pointer della routine di servizio ed è caricato con il valore X'0F1B. Riferendosi alla mappa di memoria, il confine della memoria ROM/RAM era stato scelto in termini di X'0F00, indirizzo vicino al contenuto del pointer P3. Quando il pointer è scambiato con il program counter a causa di una interruzione, lo stato è salvato in RAM utilizzando il metodo di indirizzamento relativo al program counter. Alla fine del servizio dell'interruzione, lo stato è ripristinato utilizzando un indirizzamento relativo al program counter.
In relazione alla scelta del valore di P3 quando è in esecuzione il programma principale, P3 può essere utilizzato come pointer per una zona di memorizzazione temporanea (scratch pad) dal programma principale tramite un indirizzamento indicizzato. L'area disponibile come scratch pad nell'esempio è l'area tra la fine dell'area di salvataggio (X'0EF6) e -127 dal valore presente nel pointer (X'0F1B-7F) o X'0E9C. Il pointer P3 può essere utilizzato per accedere a costanti poste in ROM alla fine della routine di start dell'interrupt tramite un metodo di indirizzamento indicizzato. L'area disponibile per la memorizzazione delle costanti va da X'0F40 a X'0F9A (X'0F1B+7F).

4-3 Tempo di risposta all'interrupt

Sia per l'interrupt a singolo livello che per il caso multi-level, il tempo di risposta all'interrupt (IRT) è un parametro molto importante. Come mostrato nella fig. 4-1, un interrupt può verificarsi con l'input CONT alto e lo SC/MP in funzione; oppure con CONT basso e lo SC/MP in halt mode. Il tempo massimo di risposta all'interrupt quando lo SC/MP sta eseguendo un'istruzione del programma principale, è eguale (nel caso peggiore) al tempo massimo di esecuzione+overhead. Il tempo di esecuzione per una istruzione di DAD è di 23 cicli macchina ($23x2T_x$), dove T_x è un ciclo di clock, eguale ad 1 microsecondo per una frequenza di clock di 1 megahertz. L'overhead è dell'ordine di grandezza di 14T+200 nanosecondi. Il tempo massimo di risposta all'interrupt è $46T_x+(14T_x+200$ ns); questo tempo non include l'istruzione di delay a causa della estrema variabilità di durata e non include nemmeno l'operazione di hold. In generale, il tempo di risposta all'interrupt è funzione delle istruzioni in esecuzione. Per un tempo di risposta più preciso, la configurazione del sistema può porre SC/MP in halt mode pilotando bassa la linea continue ed accettando gli interrupt abilitati. Al verificarsi di un interrupt il tempo di risposta è garantito essere $12T_x \leq T_{ir} \leq (14\ T_x + 200$ ns).

T = Tempo di risposta all'interrupt

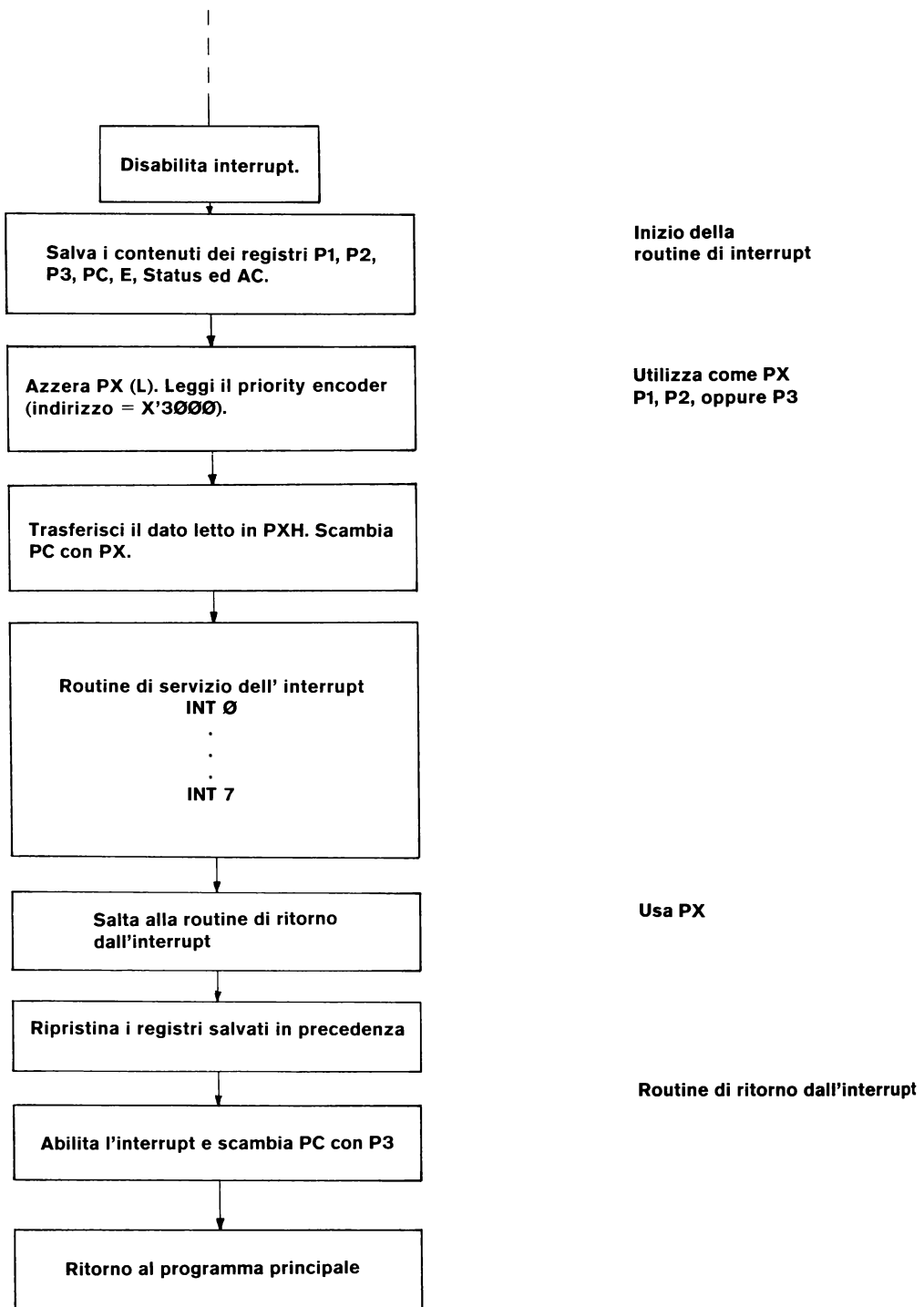



Fig. 4 - 3. Flowchart del programma di gestione di interrupt multi-level

```
END PASS 1

  1                        ;
  2                                . TITLE INTRPT, 'VECTORED INTERRUPT FOR SC/MP'
  3                        ;
  4                        ;
  5          0001                  P1      =1
  6          0002                  P2      =2
  7          0003                  P3      =3
  8                        ;
  9                        ;
 10   0000   C41B  MAIN:           LDI     L (INTRPT)-1    ; IN P3 PUNTAT. INTERRUPT
 11   0002   33                    XPAL    P3
 12   0003   C40B                  LDI     H (INTRPT)
 13   0005   37                    XPAH    P3
 14   0006   05                    IEN                     ; ABILITA INTER.
 15                        ;
 16                                ; IL PROGRAMMA UTENTE PRINCIPALE
 17                                ; INIZIA DA QUESTO PUNTO
 18                        ;
 19                        ;
 20          0EF6                  .= 0EF6
 21                        ;
 22          0EF7  STATUS:         .=.+1
 23          0EF8  EXT:            .=.+1
 24          0EF9  ACU:            .=.+1
 25          0EFA  P2H:            .=.+1
 26          0EFB  P2L:            .=.+1
 27          0EFC  P1H:            .=.+1
 28          0EFD  P1L:            .=.+1
 29          0EFE  PCH:            .=.+1
 30          0EFF  PCL:            .=.+1
 31                        ;
 32
```

```
INTRPT VECTORED INTERRUPT FOR SC/MP
INTERRUPT RETURN

                                .PAGE  'INTERRUPT RETURN'
 33          0F00               .= 0F00
 34                        ;
 35                                ; ROUTINE DI RITORNO DALL'INTERRUPT.
 36                                ; RIPRISTINA LO STATO DEL SISTEMA E
 37                                ; RITORNA AL PROGRAMMA PRICIPALE.
 38
 39   0F00   C0F6  INTRN:          LD      EXT             ; RIPRISTINA REG. E
 40   0F02   01                    XAE
 41   0F03   C0F8                  LD      P1L             ; RIPRISTINA P1
 42   0F05   31                    XPAL    P1
 43   0F06   C0F4                  LD      P1H
 44   0F08   35                    XPAH    P1
 45   0F09   C0F0                  LD      P2L             ; RIPRISTINA P2
 46   0F0B   32                    XPAL    P2
 47   0F0C   C0EC                  LD      P2H
 48   0F0E   36                    XPAH    P2
 49   0F0F   C0EE                  LD      PCL             ; RIPR. INDIRIZZO
 50   0F11   33                    XPAL    P3              ; DI RITORNO
 51   0F12   C0EA                  LD      PCH
 52   0F14   37                    XPAH    P3
 53   0F15   C0E0                  LD      STATUS          ; RIPR. REG. STATUS
 54   0F17   07                    CAS
 55   0F18   C0DF                  LD      ACU             ; RIPRISTINA AC
 56   0F1A   05                    IEN
 57   0F1B   3F                    XPPC    P3
 58                        ;
 59                        ;
 60                        ;
```

Fig. 4 - 3 Listing del programma di gestione di interrupt multilevel.

segue

Fig. 4 - 3

```
INTEPTVECTORED INTERRUPT FOR SC/MP
INTERRUPT ROUTINE

                                 .PAGE 'INTERRUPT ROUTINE'
  61
  62                             ; ROUTINE DI INTERRUPT. LEGGE IL
  63                             ; "PRIORITY ENCODER" PER
  64                             ; DETERMINARE QUALE INTERRUPT
  65                             ; RICHIEDE IL SERVIZIO E QUINDI VI
  66                             ; SALTA.
  67                    ;
  68                    ;
  69  0F1C  C8DB  INTRPT:  ST    ACU        ; SALVA AC
  70  0F1E  06             CSA
  71  0F1F  C8D6           ST    STATUS     ; SALVA REG. STATUS
  72  0F21  33             XPAL  P3         ; SALVA IND. RITORNO
  73  0F22  C8DB           ST    PCL
  74  0F24  37             XPAH  P3
  75  0F25  C8D7           ST    PCH
  76  0F27  31             XPAL  P1         ; SALVA P1
  77  0F28  C8D3           ST    P1L
  78  0F2A  35             XPAH  P1
  79  0F2B  C8CF           ST    P1H
  80  0F2D  32             XPAL  P2         ; SALVA P2
  81  0F2E  C8CB           ST    P2L
  82  0F30  36             XPAH  P2
  83  0F31  C8C7           ST    P2H
  84  0F33  01             XAE
  85  0F34  C8C2           ST    EXT        ; SALVA REG. E
  86  0F36  C400  RDSTAT:  LDI   0          ; AZZ. P2
  87  0F38  32             XPAL  P2
  88  0F39  C430           LDI   030        ; IN P2 INDIRIZZO
  89  0F3B  36             XPAH  P2         ; PRIORITY ENCODER
  90  0F3C  C200           LD    (P2)       ; LEGGI PRIORITÀ
  91  0F3E  36             XPAH  P2         ; IN P2H PUNTAT. SERVIZIO
  92  0F3F  3E             XPPC  P2         ; VAI ALLA ROUT. SERVIZIO
  93                    ;
  94                    ;
  95                             ; RITORNO DALLA ROUTINE DI SERVIZIO.
  96                             ; È FATTO ESEGUENDO L'ISTRUZIONE
  97                             ; "XPPC P2"
  98                    ;
  99  0F40  90BE  SRETRN:  JMP   INTRN      ; RITORNO DAL SERV.
 100                                        ; VAI ALLA ROUTINE DI
 101                                        ; RITORNO DALL'INTER.
 102                    ;
 103        0000           .END

ACU      0EF8      EXT      0EF7      INTRN    0F00
INTRPT   0F1C      MAIN     0000*     P1       0001
P1H      0EFB      P1L      0EFC      P2       0002
P2H      0EF9      P2L      0EFA      P3       0003
PCH      0EFD      PCL      0EFE      RDSTAT   0F36*
SRETRN   0F40*     STATUS   0EF6

ERROR LINES
END PASS 2
SOURCE CHECKSUM = DF66

NEXT ASSEMBLY
*.ASM
```

Capitolo 5°

ROUTINE MATEMATICHE

Tramite il set di istruzioni del microprocessore SC/MP è possibile implementare le routine matematiche indicate in questo capitolo. Le routine sono allocate in modo continuo in memoria, tuttavia una qualunque routine può essere utilizzata come programma a sé stante nell'ambito di una specifica applicazione. Le routine matematiche presentate sono le seguenti:

- Somma in doppia precisione
- Negazione in doppia precisione
- Sottrazione in doppia precisione
- Moltiplicazione senza segno
- Moltiplicazione con segno
- Divisione senza segno
- Moltiplicazione BCD
- Somma e sottrazione BCD
- Complemento BCD
- Divisione BCD

```
 1          TITLE MATH
 2   ;      ROUTINE MATEMATICHE PER LO SC/MP
 3   ;
 4   ;       TUTTE LE ROUTINE LAVORANO CON LO STACK
 5   ;
 6   ;      GLI OPERANDI SONO PRESI DALLO STACK ED
 7   ;      I RISULTATI SONO POSTI NELLO STACK
 8   ;
 9   ;      IN TUTTI GLI ESEMPI, OP1 SI RIFERISCE AL
10   ;      BYTE AL TOP DELLO STACK, RELATIVAMENTE AL
11   ;      POINTER ORIGINALE;
12   ;      OP2 È IL BYTE SUCCESSIVO E COSÌ VIA
13   ;
14   ;      ESEMPIO DI UTILIZZO DELLO STACK PER UNA
15   ;      DOUBLE-ADD:
16   ;      DADD:  (OP3, OP4) = (OP1, OP2) + (OP3, OP4)
17   ;      LO STACK È DA 0000 A 001F
18   ;
19
20   ;      001B   OP1   (HIGH1) <-- STACK POINTER IN INGRESSO
21   ;      001C   OP2   (LOW1)
22   ;      001D   OP3   (HIGH2) <-- STACK POINTER IN USCITA
23   ;      001E   OP4   (LOW2)
24   ;      001F   (FINE DELLO STACK)
25   ;
26   ;
27   ;
28   ;      I REGISTRI A ED E (E P1 SE È UTILIZZATO)
29   ;      NON SONO SALVATI
30   ;      IL REGISTRO P2 È UTILIZZATO COME STACK POINT
31   ;      IL REGISTRO P3 È UTILIZZATO COME SUBROUTINES
32   ;      POINTER
33   ;
34   ;
35   ;      TUTTE LE ROUTINE MATAMATICHE POSSONO ESSERE
36   ;      RIPETUTE SENZA REINIZIALIZZARE P3 DOPO LA PRIMA
37   ;      CHIAMATA DI SUBROUTINE
```

MATH
DADD, DNEG, DSUB

```
                                  . PAGE  'DADD, DNEG, DSUB'
38                        ;         QUESTA ROUTINE ATTUA UNA DOUBLE ADD, UNA DOUBLE
39                        ;         NEGATE ED UNA DOUBLE SUBTRACT
40                        ;
41          1000                  .= 01000
42                        ;
43                        ;         DADD: DOUBLE ADD
44                        ;         (OP3, OP4) = (OP1, OP2) + (OP3, OP4)
45                        ;
46   1000   08           DADD:    NOP
47   1001   02                    CCL
48   1002   C201                  LD       1(2)         ; SOMMA IL LOW ORDER BYTE
49   1004   F203                  ADD      3(2)         ; (IL CARRY PUÒ
50                                                      ; ESSERE SETTATO)
51   1006   CA03                  ST       3(2)         ; SALVA IL RISULTATO
52   1008   C200                  LD       0(2)         ; SOMMA IL HIGH ORDER
53                                                      ; BYTE+ CARRY
54   100A   F202                  ADD      2(2)
55   100C   CA02                  ST       2(2)
56   100E   C602                  LD       @ 2(2)
57   1010   3F                    XPPC     3
58   1011   90ED                  JMP      DADD         ; SALTO NEL CASO DI
59                                                      ; RICHIAMO
60                        ;
61                        ;*****************************************************************
62                        ;
63                        ;         DNEG: DOUBLE NEGATE (COMPLEMENTO A 2)
64                        ;
65   1013   08           DNEG:    NOP
66   1014   03                    SCL                   ; SET CARRY IN
67   1015   C400                  LDI      0
68   1017   FA01                  CAD      1(2)         ; NEGAZIONE DEL LOW
69                                                      ; BYTE E SET DEL CARRY
70   1019   CA01                  ST       1(2)
71   101B   C400                  LDI      0            ; COMPLEMENTO DELL'HIGH
72   101D   FA00                  CAD      0(2)         ; BYTE E SOMMA DEL CAR-
73                                                      ; RY IN
74   101F   CA00                  ST       0(2)
75   1021   3F                    XPPC     3
76   1022   90EF                  JMP      DNEG
77                        ;
78                        ;*****************************************************************
79                        ;
80                        ;       DSUB: DOUBLE SUBTRACT
81                        ;       (OP3, OP4) = (OP3, OP4) - (OP1, OP2)
82                        ;
83   1024   08           DSUB:    NOP
84   1025   03                    SCL                   ; SET DEL CARRY IN
85                                                      ; (BORRO W)
86   1026   C203                  LD       3(2)         ; SOMMA IL LOW BYTE
87   1028   FA01                  CAD      1(2)         ; OP4 + (-OP2)
88   102A   CA03                  ST       3(2)
89   102C   C202                  LD       2(2)
90   102E   FA00                  CAD      0(2)
91   1030   CA02                  ST       2(2)
92   1032   C602                  LD       @ 2(2)
93   1034   3F                    XPPC     3
94   1035   90ED                  JMP      DSUB
95                        ;
```

MATH
UNSIGNED MULTIPLY

```
                                  .PAGE 'UNSIGNED MULTIPLY'
96                                .LOCAL
97                        ;
```

```
 98                   ;
 99                   ;         MOLTIPLICAZIONE SENZA SEGNO
100                   ;         (OP1, OP2) = (OP1) * (OP2)
101                   ;
102                   ;         UN BYTE DELLO STACK È UTILIZZATO COME
103                   ;         ZONA DI MEMORIZZAZIONE TEMPORANEA:
104                   ;         -1(2) = CONTATORE DI BIT
105                   ;
106                   ;         ; TEMPO DI ESECUZIONE: 2,5 MILLISECONDI C.A.
107                   ;         ; TEMPO MASSIMO: 3 MILLISECONDI
108                   ;
109  1037  08         MPY:      NOP
110  1038  C200                 LD      0(2)        ; PRENDI IL MOLTIPLICATORE
111  103A  01                   XAE                 ; SALVATAGGIO DEL
112                                                 ; MOLTIPLICATORE NEL
113                                                 ; REGISTRO E
114  103B  C400                 LDI     0
115  103D  CA00                 ST      0(2)
116  103F  C408                 LDI     8
117  1041  CAFF                 ST      -1(2)       ; SALVATAGGIO DEL
118                                                 ; COUNTER
119  1043  40         $LOOP:    LDE                 ; CARICAMENTO DEL
120                                                 ; MOLTIPLICATORE
121  1044  D401                 ANI     1           ; TEST DEL LEAST BIT
122  1046  9815                 JZ      NO          ; IL BIT NON È 1
123  1048  C200                 LD      0(2)        ; SOMMA IL MOLTIPLICANDO
124  104A  F201                 ADD     1(2)        ; (8 BIT) AL RISULTATO
125                                                 ; A 16 BIT
126  104C  02         NOADD:    CCL                 ; CLEAR DEL LINK
127  104D  1F                   RRL                 ; SHIFT
128  104E  CA00                 ST      0(2)
129  1050  40                   LDE
130  1051  1F                   RRL
131  1052  01                   XAE
132  1053  BAFF                 DLD     -1(2)       ; DECREMENTO DEL
133                                                 ; CONTATORE
134  1055  9CEC                 JNZ     $LOOP       ; SE IL CONTATORE È
135                                                 ; DIVERSO DA 0,CI SI
136                                                 ; COLLEGA CON $LOOP
137  1057  40                   LDE
138  1058  CA01                 ST      1(2)
139  105A  3F                   XPPC    3           ; RITORNO
140  105B  90DA                 JMP     MPY
141  105D  C200       NO:       LD      0(2)
142  105F  90EB                 JMP     NOADD
143                   ;         .PAGE  'SIGNED MULTIPLY'
144                   ;
145                   ;         SMPY: MOLTIPLICAZIONE CON SEGNO
146                   ;         (OP1, OP2) = (OP1) * (OP2)
147                   ;         OP1 ED OP2 SONO NUMERI CON SEGNO
148                   ;         IN UN RANGE DA -128 A +127
149                   ;
150  1061  08         SMPY:     NOP
151  1062  C410                 LDI     H(MPY)      ; SI CARICA IN P3 IL VA-
152                                                 ; LORE PER CHIAMARE MPY
153  1064  37                   XPAH    3           ; SALVATAGGIO DEL P3
154  1065  35                   XPAH    1           ; ORIGINALE IN P1
155  1066  C437                 LDI     L(MPY)
156  1068  33                   XPAL    3
157  1069  31                   XPAL    1
158  106A  02                   CCL
159  106B  C200                 LD      0(2)        ; INIZIO CONFRONTO
160                                                 ; DEI SEGNI
161  106D  E201                 XOR     1(2)
162  106F  9404                 JP      $SAME
163  1071  C4FF                 LDI     -1
164  1073  9002                 JMP     $MPY
```

```
165  1075  C400  $SAME:  LDI   0
166  1077  CAFE  $MPY:   ST    -2(2)
167  1079  C200          LD    0(2)        ; CHECK DEL SEGNO
168  107B  9407          JP    $MPY2
169  107D  03            SCL
170  107E  C400          LDI   0
171  1080  FA00          CAD   0(2)        ; NEGAZIONE DEL PRIMO
172                                        ; OPERANDO
173  1082  CA00          ST    0(2)
174  1084  C201  $MPY2:  LD    1(2)
175  1086  9007          JMP   $MPY3
176  1088  03            SCL
177  1089  C400          LDI   0
178  108B  FA01          CAD   1(2)
179  108D  CA01          ST    1(2)
180  108F  3F     MPY3:  XPPC  3           ; ATTUAZIONE DELLA MOLTIPLI-
181                                        ; CAZIONE SENZA SEGNO
182  1090  C2FE          LD    -2(2)       ; CHECK DEL FLAG DI SEGNO
183  1092  940D          JP    $MPY4
184  1094  03            SCL
185  1095  C400          LDI   0
186  1097  FA01          CAD   1(2)        ; NEGAZIONE DEL RISULTATO
187  1099  CA01          ST    1(2)
188  109B  C400          LDI   0
189  109D  FA00          CAD   0(2)
190  109F  CA00          ST    0(2)
191  10A1  35    $MPY4:  XPAH  1           ; RIPRISTINO DELLO
192  10A2  37            XPAH  3           ; INDIRIZZO DI RITORNO
193  10A3  31            XPAL  1           ; DA P1
194  10A4  33            XPAL  3
195  10A5  3F            XPPC  3
196  10A6  90B9          JMP   $MPY
197                      .PAGE "USIGNED DIVIDE"
198                      .LOCAL
199                 ;
200                 ;      DIV: DIVISIONE SENZA SEGNO
201                 ;
202                 ;      VIENE EFFETTUATA LA DIVISIONE TRA UN NUMERO
203                 ;      A 16 BIT PRIVO DI SEGNO ED UN NUMERO AD 8
204                 ;      BIT ANCH'ESSO PRIVO DI SEGNO
205                 ;      IL RISULTATO È UN QUOZIENTE A 16 BIT ED UN
206                 ;      RESTO AD 8 BIT
207                 ;
208                 ;      TEMPO DI ESECUZIONE: DA 5 A 20 MILLISECONDI
209                 ;
210                 ;      CHIAMATA: XPPC3
211                 ;                (RITORNO CON ERRORE)
212                 ;                (RITORNO NORMALE)
213                 ;
214                 ;      UTILIZZO DELLO STACK:
215                 ;      REL. ENTRY          UTILIZZO        RITORNO
-------------------------------------------------------------------------
217                 ;      -5                  TEMP
218                 ;      -4                  CONTATORE
219                 ;      -3                  RESTO (H)
220                 ;      -2                  RESTO (L)
221                 ;      -1                  DIVISORE (H)
222                 ;(P2)->  0  DIVISORE       DIVISORE (L)    RESTO
223                 ;        1  DIVIDENDO (H)  QUOZ. (H)       QUOZ.
224                 ;        2  DIVIDENDO (L)  QUOZ. (L)       QUOZ.
225                 ;
226                 ;
227  10A8  08    DIV:    NOP
228  10A9  C200          LD    0(2)        ; TEST SE IL DIVISORE
229  10AB  9C03          JNZ   $D1         ; È ZERO
230  10AD  3F            XPPC  3           ; RITORNO PER ERRORE
231  10AE  90F8          JMP   DIV
232  10B0  C409  $D1:    LDI   9           ; CONTATORE = 9
```

```
233  10B2  CAFC              ST    -4(2)
234  10B4  C200              LD    0(2)
235  10B6  01                XAE
236  10B7  40     $SET:      LDE               ; NORMALIZZAZIONE DEL
237                                            ; DIVISORE
238  10B8  9402              JP    .+ 4
239  10BA  9007              JMP   $SETUP
240  10BC  02                CCL               ; SHIFT A SINISTRA DI
241                                            ; UN BIT
242  10BD  70                ADE
243  10BE  01                XAE
244  10BF  AAFC              ILD   -4(2)       ; CONTATORE = CONT+1
245  10C1  90F4              JMP   $SET
246  10C3  40     $SETUP:    LDE
247  10C4  CAFF              ST    -1(2)       ; SALVATAGGIO DIVISORE
248  10C6  C201              LD    1(2)
249  10C8  CAFD              ST    -3(2)       ; COPIA DEL DIVIDENDO
250  10CA  C202              LD    2(2)        ; NEL RESTO INIZIALE
251  10CC  CAFE              ST    -2(2)
252  10CE  C400              LDI   0
253  10D0  CA00              ST    0(2)
254  10D2  CA01              ST    1(2)
255  10D4  CA02              ST    2(2)
256  10D6  03     $LOOP:     SCL
257  10D7  C2FE              LD    -2(2)       ; SOTTRAZIONE:
258  10D9  FA00              CAD   0(2)        ; RESTO-DIVISORE
259  10DB  01                XAE
260  10DC  C2FD              LD    -3(2)
261  10DE  FAFF              CAD   -1(2)
262  10E0  CAFB              ST    -5(2)       ; SALVATAGGIO TEMP.
263  10E2  06                CSA               ; TEST DEL CARRY
264  10E3  E480              XRI   080
265  10E5  941E              JP    $DVGR       ; JUMP SE IL RISULTATO
266                                            ; È MAGGIORE OD
267                                            ; EGUALE A ZERO
268  10E7  BAFC   DLD        -4(2) CONTATORE   = CONT.-1
269  10E9  9829              JZ    $DONE
270  10EB  02                CCL
271  10EC  C202              LD    2(2)        ; DOPPIO SHIFT A SIN.
272  10EE  F202              ADD   2(2)        ; SOMMA IL QUOZIENTE A
273                                            ; SÈ STESSO
274  10F0  CA02              ST    2(2)
275  10F2  C201              LD    1(2)
276  10F4  F201              ADD   1(2)
277  10F6  CA01              ST    1(2)
278  10F8  02                CCL
279  10F9  C2FF              LD    -1(2)       ; SHIFT DEL DIVISORE A
280  10FB  1F                RRL               ; DESTRA DI 1 BIT
281  10FC  CAFF              ST    -1(2)
282  10FE  C200              LD    0(2)
283  1100  1F                RRL
284  1101  CA00              ST    0(2)
285  1103  90D1              JMP   $LOOP
286  1105  C2FB   $DVGR:     LD    -5(2)       ; SALVA IL NUOVO RESTO
287  1107  CAFD              ST    -3(2)       ; SEGUENTE LA SOTTRAZIONE
288  1109  01                XAE
289  110A  CAFE              ST    -2(2)
290  110C  AA02              ILD   2(2)        ; INCREMENTO QUOZ.
291  110E  9CC6              JNZ   $LOOP
292  1110  AA01              ILD   1(2)
293  1112  90C2              JMP   $LOOP
294  1114  C2FE   $DONE:     LD    -2(2)       ; COPIA DEL RESTO FINALE
295  1116  CA00              ST    0(2)
296  1118  C702              LD    @2(3)       ; INCREMENTO DI P3 PER
297                                            ; IL RITORNO NORMALE
298  111A  3F                XPPC  3
299  111B  908B              JMP   DIV
300                ;
301
```

```
                                   .PAGE 'BCD MULTIPLY'
302                     ;          .LOCAL
303                     ;
304                     ;
305                     ;           BCDMPY: ROUTINE DI MOLTIPLICAZIONE
306                     ;           DI DUE NUMERI BCD A 6 DIGIT CON RISULTATO
307                     ;           ESPRESSO SU 12 DIGIT
308                     ;
309                     ;           TEMPO DI ESECUZIONE: DA 6 A 27 MILLISECONDI
310                     ;           CON UN TEMPO TIPICO DI 17 MILLISECONDI
311                     ;
312                     ;          ; UTILIZZO DELLO STACK:
313                     ;          REL    ENTRY          USE              RETURN
314                     ;          ------------------------------------------------------
315                     ;          -10                   TEMP
316                     ;          -9                    PRODUCT (1)
317                     ;          -8                    PRODUCT (2)
318                     ;          -7                    PRODUCT (3)
319                     ;          -6                    PRODUCT (4)
320                     ;          -5                    PRODUCT (5)
321                     ;          -4                    PRODUCT (6)
322                     ;          -3                    COUNT
323                     ;          -2                    M1 (1)
324                     ;          -1                    M1 (2)
325                     ;(P2)->     0     M1 (SIGN)      M1 (3)
326                     ;           1     M1 (1)         M1 (4)           PRODUCT (SIGN)
327                     ;           2     M1 (2)         M1 (5)           PRODUCT (1)
328                     ;           3     M1 (3)         M1 (6)           PRODUCT (2)
329                     ;           4     M2 (SIGN)      PROD (SIGN)      PRODUCT (3)
330                     ;           5     M2 (1)         M2 (1)           PRODUCT (4)
331                     ;           6     M2 (2)         M2 (2)           PRODUCT (5)
332                     ;           7     M2 (3)         M2 (3)           PRODUCT (6)
333                     ;
334                     ;
335         FFFE                   M1      =-2            ; MOLTIPLICANDO
336                                                       ; (6 PAROLE)
337         0005                   M2      =5             ; MOLTIPLICATORE
338                                                       ; (3 PAROLE)
339         FFF7                   PR      =-9            ; PRODOTTO
340                                                       ; (6 PAROLE)
341         FFFD                   COUNT   =-3            ; CONTATORE DI DIGIT
342                                                       ; (1 PAROLA)
343         FFF6                   TEMP    =-10           ; SHIFT COUNT
344         FFF6                   M3      =-10           ; DIGIT MOLTIPLICATORE
345                     ;
346   111D  08          BCDMPY:    NOP
347   111E  C200                   LD      0(2)           ; CALCOLO DEL
348                                                       ; SEGNO DEL PRODOTTO
349   1120  02                     CCL
350   1121  F204                   ADD     4(2)
351   1123  D401                   ANI     1
352   1125  CA04                   ST      4(2)           ; SALVATAGGIO DEL SEGN
353                                                       ; DEL PRODOTTO
354   1127  C400                   LDI     0              ; INIZIO PRODOTTO
355   1129  CAF7                   ST      PR(2)
356   112B  CAF8                   ST      PR+1(2)
357   112D  CAF9                   ST      PR+2(2)
358   112F  CAFA                   ST      PR+3(2)
359   1131  CAFB                   ST      PR+4(2)
360   1133  CAFC                   ST      PR+5(2)
361   1135  CAFE                   ST      M1(2)
362   1137  CAFF                   ST      M1+1(2)
363   1139  CA00                   ST      M1+2(2)
364   113B  C406                   LDI     6              ; CONTATORE  DEI  DIGIT
365                                                       ; EGUALE A 6
366   113D  CAFD                   ST      COUNT(2)
```

```
367  113F  C207  $LOOP:  LD    M2+2(2)     ; CARICAMENTO DEL
368                                        ; MOLTIPLICA-
369                                        ; TORE
370  1141  D40F          ANI   0F          ; VIENE UTILIZZATO IL
371                                        ; DIGIT BCD PIÙ LEGGE
372  1143  982B          JZ    NEXTD       ; SE È 0, SI PASSA AL
373                                        ; DIGIT SUCCESSIVO
374  1145  CAF6          ST    M3(2)
375  1147  02    $L1:    CCL               ; SOMMA IL MOLTIPLICANDO
376                                        ; AL PRODOTTO
377  1148  C203          LD    M1+5(2)
378  114A  EAFC          DAD   PR+5(2)
379  114C  CAFC          ST    PR+5(2)
380  114E  C202          LD    M1+4(2)
381  1150  EAFB          DAD   PR+4(2)
382  1152  CAFB          ST    PR+4(2)
383  1154  C201          LD    M1+3(2)
384  1156  EAFA          DAD   PR+3(2)
385  1158  CAFA          ST    PR+3(2)
386  115A  C200          LD    M1+2(2)
387  115C  EAF9          DAD   PR+2(2)
388  115E  CAF9          ST    PR+2(2)
389  1160  C2FF          LD    M1+1(2)
390  1162  EAF8          DAD   PR+1(2)
391  1164  CAF8          ST    PR+1(2)
392  1166  C2FE          LD    M1(2)
393  1168  EAF7          DAD   PR(2)
394  116A  CAF7          ST    PR(2)
395  1160  BAF6          DLD   M3(2)
396  116C                                  ; DECREMENTO DEL
397                                        ; MOLTIPLICATORE
398  116E  9CD7          JNZ   $L1         ; NUOVA SOMMA SE DIVERSA
399                                        ; DA ZERO
400  1170  BAFD  NEXTD:  DLD   COUNT(2)    ; DECREMENTO DEL CONTA-
401                                        ; TORE DI DIGIT
402  1172  9847          JZ    $OUT
403  1174  C404          LDI   4           ; SHIFT DEL MOLTIPLI-
404  1176  CAF6          ST    TEMP(2)     ; CANDO A SINISTRA DI
405                                        ; 4 BIT (1 DIGIT)
406  1178  02    $L2:    CCL
407  1179  C203          LD    M1+5(2)
408  117B  F203          ADD   M1+5(2)
409  117D  CA03          ST    M1+5(2)
410  117F  C202          LD    M1+4(2)
411  1181  F202          ADD   M1+4(2)
412  1183  CA02          ST    M1+4(2)
413  1185  C201          LD    M1+3(2)
414  1187  F201          ADD   M1+3(2)
415  1189  CA01          ST    M1+3(2)
416  118B  C200          LD    M1+2(2)
417  118D  F200          ADD   M1+2(2)
418  118F  CA00          ST    M1+2(2)
419  1191  C2FF          LD    M1+1(2)
420  1193  F2FF          ADD   M1+1(2)
421  1195  CAFF          ST    M1+1(2)
422        C2FE          LD    M1(2)
423  1119  F2F3          ADD   M1(2)
424  119B  CAFE          ST    M1(2)
425  119D  BAF6          DLD   TEMP(2)
426  119F  9CD7          JNZ   $L2
427  11A1  C2FD          LD    COUNT(2)
428                                        ; PRENDI IL NUOVO DIGIT
429                                        ; DEL MOLTIPLICATORE
430  11A3  D401          ANI   1           ; SE IL COUNT È 'EVEN
431                                        ; UTILIZZA IL SUCC.
432  11A5  980A          JZ    $L3
433  11A7  C207          LD    M2+2(2)     ; ALTRIMENTI EFFETTUA
434                                        ; UNO SHIFT A DESTRA
435                                        ; DELLA PAROLA DI ORDINE
```

```
                                                      ; PIÙ BASSO
437  11A9  1C              SR
438  11AA  1C              SR
439  11AB  1C              SR
440  11AC  1C              SR
441  11AD  CA07            ST     M2+2(2)
442  11AF  9090            JMP    $LOOP
443  11B1  C206   $L3:     LD     M2+1(2)
444  11B3  CA07            ST     M2+2(2)
445  11B5  C205            LD     M2(2)
446  11B7  CA06            ST     M2+1(2)
447  11B9  C207            LD     M2+2(2)
448  11BB  9084            JMP    $LOOP
449  11BD  C204   $OUT:    LD     4(2)
450  11BF  CA01            ST     1(2)
451  11C1  C2F7            LD     PR(2)
452  11C3  CA02            ST     2(2)
453  11C5  C2F8            LD     PR+1(2)
454  11C7  CA03            ST     3(2)
455  11C9  C2F9            LD     PR+2(2)
456  11CB  CA04            ST     4(2)
457  11CD  C2FA            LD     PR+3(2)
458  11CF  CA05            ST     5(2)
459  11D1  C2FB            LD     PR+4(2)
460  11D3  CA06            ST     6(2)
461  11D5  C2FC            LD     PR+5(2)
462  11D7  CA07            ST     7(2)
463  11D9  3F              XPPC   3           ; RITORNO
464  11DA  C411            LDI    H(BCDMPY)   ; SALTO ALLA BCDMPY
465  11DC  35              XPAH   1
466  11DD  C41D            LDI    L(BCDMPY)
467  11DF  31              XPAL   1
468  11E0  91FF            JMP    0(1)
469                 ;
470
```

MATH
BCD ADD/SUBTRACT

```
                           .PAGE  'BCD ADD/SUBTRACT'
471                        ,LOCAL
472                 ;
```

MATH
BCD ADD/SUBTRACT

```
                           .PAGE  'BCD ADD/SUBTRACT'
471                        .LOCAL
472                 ;
473                 ;        ADDIZIONE E SOTTRAZIONE BCD
474                 ;        OGNI OPERANDO È 6 DIGIT
475                 ;        PIÙ IL SEGNO
476                 ;
477                 ;        TEMPO DI ESECUZIONE: DA 630 A 1110 MSEC.
478                 ;
479                 ;        BCDADD: OP2 = OP2+OP1
480                 ;        BCDSUB: OP2 = OP2—OP1
481                 ;
482                 ;        OP1 NON VIENE MODIFICATO
483                 ;
484                 ;        SEGNO: 0 = POS.    1 = NEG.
485                 ;
486                 ;      ; CALL: XPPC 3
487                 ;
```

```
488                 ;         ; UTILIZZO DELLO STACK:
489                 ;         REL  ENTRY         RITORNO
490                 ;         ------------------------------------
491                 ;         0    OP1 (SIGN)    OP1 (SIGN)
492                 ;         1    OP1 (1)       OP1 (1)
493                 ;         2    OP1 (2)       OP1 (2)
494                 ;         3    OP1 (3)       OP1 (3)
495                 ;         4    OP2 (SIGN)    RESULT (SIGN)
496                 ;         5    OP2 (1)       RESULT (1)
497                 ;         6    OP2 (2)       RESULT (2)
498                 ;         7    OP2 (3)       RESULT (3)
499                 ;
500        0001               $OP1     =1           ; PRIMO OPERANDO
501        0005               $OP2     =5           ; SECONDO OPERANDO
502                 ;
503  11E1  C201     BCDSUB:   LD       $OP1 (2)     ; TEST SE OP1 = 0
504  11E3  9C08               JNZ      $BSUB        ; SALTO SE NON 0
505  11E5  C202               LD       $OP1+1(2)    ; TEST SE OP1 = 0
506  11E7  9C04               JNZ      $BSUB        ; SALTO SE NON 0
507  11E9  C203               LD       $OP1+2(2)    ; TEST SE OP1 = 0
508  11EB  9827               JZ       $OUT         ; RITORNO SE OP1 = 0
509  11ED  C200     $BSUB:    LD       $OP1−1(2)    ; CAMBIO DEL SEGNO
510                                                 ; DEL SOTTRAENDO
511  11EF  E401               XRI      01           ; SUCCESSIVAMENTE SI
512  11F1  01                 XAE                   ; ESEGUE LA SOTTRAZIONE
513  11F2  9003               JMP      $CHK
514  11F4  C200     BCDADD:   LD       $OP1-1(2)    ; COMPARAZIONE TRA I
515                                                 ; SEGNI
516  11F6  01                 XAE
517  11E7  C204     $CHK:     LD       $OP2—1 (2)
518  11F9  60                 XRE
519  11FA  9C21               JNZ      $DIFF        ; I SEGNI SONO DIFFERENTI
520                                                 ; SALTO
521  11FC  02       SAME:     CCL                   ; STESSI SEGNI
522  11FD  C203               LD       $OP1+2(2)    ; SOMMA GRANDEZZE
523  11FF  EA07               DAD      $OP2+2(2)
524  1201  CA07               ST       $OP2+2(2)
525  1203  C202               LD       $OP1+1(2)
526  1205  EA06               DAD      $OP2+1(2)
527  1207  CA06               ST       $OP2+1(2)
528  1209  C201               LD       $OP1 (2)
529  120B  EA05               DAD      $OP2 (2)
530  120D  CA05               ST       $OP2(2)
531  120F  06                 CSA                   ; TEST SE È PRESENTE
532                                                 ; UN CARRY FINALE
533  1210  D480               ANI      080
534  1212  9C02               JNZ      $OVFL        ; È PRESENTE, QUINDI
535                                                 ; OVERFLOW
536  1214  C702     $OUT:     LD       @ 2(3)       ; INCREMENTO DI P3 DI
537                                                 ; 2 PER UN RITORNO NORMALE
538  1216  3F       $OVFL:    XPPC     3            ; RITORNO
539  1217  C401     OP2Z:     LDI      1            ; SI PONE NEGATIVO IL SEGNO
540                                                 ; DI OP2
541  1219  CA04               ST       $OP2—1(2)
542  121B  90DF               JMP      SAME
543                 ;
544  121D  40       $DIFF:    LDE                   ; SE OP1 È NEGATIVO
545                                                 ; ALLORA CI SI METTE
546                                                 ; NELLE CONDIZIONI DI
547                                                 ; AVERE:
548                                                 ; OP2—OP1=—(OP1—OP2)
549  121E  9802               JZ       $2           ; SALTO SE OP1 (SIGN) = 0
550  1220  C401               LDI      1
551  1222  CA04     $2:       ST       $OP2-1(2)    ; IL FLAG DI SEGNO
552                                                 ; È IN SIGN
553  1224  C205               LD       $OP2 (2)     ; TEST SE OP2 = 0
554  1226  9C08               JNZ      OK           ; NON LO È
```

```
555  1228  C206           LD    $OP2+1(2)
556  122A  9C04           JNZ   OK
557  122C  C207           LD    $OP2+2(2)
558  122E  98E7           JZ    OP2Z        ; OP2 = 0
559  1230  C605   OK:     LD    @ $OP2 (2)  ; SET DI P2 PER
560                                         ; PUNTARE SU OP2
561  1232  C412           LDI   H(BCDCMP)   ; COMPLEMENTO DI OP2
562  1234  35             XPAH  1
563  1235  C47C           LDI   L(BCDCMP)
564  1237  31             XPAL  1
565  1238  02             CCL
566  1239  3D             XPPC  1
567  123A  C6FB           LD      @ $OP2 (2) ; RIPRISTINO DI P2
568  123C  02             CCL
569  123D  C203           LD    $OP1+2(2)
570  123F  EA07           DAD   $OP2+2(2)
571  1241  CA07           ST    $OP2+2(2)
572  1243  C202           LD    $OP1+1(2)
573  1245  EA06           DAD   $OP2+1(2)
574  1247  CA06           ST    $OP2+1(2)
575  1249  C201           LD    $OP1(2)
576  124B  EA05           DAD   $OP2(2)
577  124D  CA05           ST    $OP2(2)
578  124F  06             CSA               ; TEST SE CARRY
579  1250  D480           ANI   080
580  1252  9C10           JNZ   $PLUS       ; È PRESENTE UN CARRY
581                                         ; SEGNO POSITIVO
582  1254  C605           LD    @ $OP2(2)   ; SET DI P2 A RESULT
583  1256  C412           LDI   H(BCDCMP)   ; COMPLEMENTO A 10
584                                         ; DEL RISULTATO
585  1258  35             XPAH  1
586  1259  C47C           LDI   L(BCDCMP)
587  125B  31             XPAL  1
588  125C  02             CCL
589  125D  3D             XPPC  1
590  125E  C6FB           LD       $OP2 (2) ; RIPRISTINO DI P2
591  1260  C401           LDI   1           ; SEGNO NEGATIVO
592  1262  9002           JMP   $RTN2
593  1264  C400   $PLUS:  LDI   0           ; SEGNO POSITIVO
594  1266  E204   $RTN2:  XOR  @$OP2—1 (2)  ; CAMBIO DEL SEGNO SE
595                                         ; IL FLAG È SETTATO
596  1268  CA04           ST    $OP2-1(2)   ; SALVATAGGIO DEL SEGNO
587                                         ; CORRETTO DEL RISULTATO
598  126A  C205           LD    $OP2 (2)    ; TEST SE ZERO
599  126C  9CA6           JNZ   $OUT        ; RITORNO
600  126E  C206           LD    $OP2+1(2)
601  1270  9CA2           JNZ   $OUT
602  1272  C207           LD    $OP2+2(2)
603  1274  9C9E           JNZ     OUT
604  1276  C400           LDI   0
605  1278  CA04           ST    $OP2-1(2)
606  127A  9098           JMP   $OUT
607                   ;
608

MATH
BCD COMPLEMENT

                          .PAGE 'BCD COMPLEMENT'
609                       .LOCAL
610                   ;
611                   ;
612                   ;   BCDCMP: COMPLEMENTO DI UN NUMERO BCD
613                   ;   A 6 DIGIT
614                   ;
615                   ;   CALL: XPPC 1
616                   ;   P2 PUNTA AL PRIMO BYTE DEL NUMERO
617                   ;   CHE DEVE ESSERE COMPLEMENTATO
618
```

```
619                          ;         È UTILIZZATA SOLO PER LA SOMMA E LA
620                          ;         SOTTRAZIONE BCD
621                          ;
622   127C    08      BCDCMP: NOP
623   127D    C49A            LDI      09A
624   127F    01              XAE
625   1280    40              LDE
626   1281    FA02            CAD      2(2)
627   1283    CA02            ST       2(2)
628   1285    02              CCL
629   1286    40              LDE
630   1287    FA01            CAD      1(2)
631   1289    CA01            ST       1(2)
632   128B    02              CCL
633   128C    40              LDE
634   128D    FA00            CAD      0(2)
635   128F    CA00            ST       0(2)
636   1291    03              SCL
637   1292    C400            LDI      0
638   1294    EA02            DAD      2(2)
639   1296    CA02            ST       2(2)
640   1298    C400            LDI      0
641   129A    EA01            DAD      1(2)
642   129C    CA01            ST       1(2)
643   129E    C400            LDI      0
644   12A0    EA00            DAD      (2)
645   12A2    CA00            ST       (2)
646   12A4    3D              XPPC     1
647                          ;
648                          ;
649
```

MATH
BCD DIVIDE

```
                                  .PAGE 'BCD DIVIDE'
650                               .LOCAL
651                          ;
652                          ;    BCDDIV: DIVISIONE BCD
653                          ;
654                          ;      GLI OPERANDI SONO COSTITUITI DA
655                          ;    6 DIGIT PIÙ IL SEGNO
656                          ;      TEMPO DI ESECUZIONE: DA 6 A 127 MSEC.
657                          ;
658                          ;      CALL: XPPC 3
659                          ;
660           FFF8                $TEMP    =-8
661           FFF9                $CNT     =-7
662           FFFA    FFFA        $P3      =-6
663           FFFD                $Q       =-3        ; QUOZIENTE
664           0001                $DI      = 1        ; DIVISORE
665           0005                $D2      = 5        ; DIVIDENDO
666                          ;
667   12A5    08      BCDDIV:     NOP
668   12A6    C201                LD       $D1(2)     ; TEST SE DIVISORE = 0
669   12A8    9C0B                JNZ      $C1
670   12AA    C202                LD       $D1+1(2)
671   12AC    9C07                JNZ      $C1
672   12AE    C203                LD       $D1+2(2)
673   12B0    9C03                JNZ      $C1
674   12B2    3F                  XPPC     3          ; RITORNO PER ERRORE
675   12B3    90F0                JMP      BCDDIV
676   12B5    37      $C1:        XPAH     3          ; SALVATAGGIO DI P1
677   12B6    CAFA                ST       $P3(2)
678   12B8    33                  XPAL     3
679   12B9    CAFB                ST       $P3+1(2)
680   12BB    C401                LDI      1          ; CONTATORE = 1
681   12BD    CAF9                ST       $CNT(2)
```

```
682  12BF  C200           LD    0(2)
683  12C1  E204           XOR   4(2)          ; TEST SE IL SEGNO
684                                           ; È IL MEDESIMO
685  12C3  CAFC           ST    -4(2)         ; SALVATAGGIO DEL SEGNO
686  12C5  C201   SET:    LD    $D1 (2)       ; NORMALIZZAZIONE DEL
687                                           ; DIVISORE
688  12C7  D4F0           ANI   0F0           ; TEST SE IL DIGIT DI
689                                           ; ORDINE PIÙ ALTO È
690  12C9  9C1F           JNZ   $SETUP        ; NON È 0, NORM.
691  12CB  C404           LDI   4
692  12CD  CAF8           ST    $TEMP (2)
693  12CF  02     $SL1:   CCL                 ; SHIFT A SINISTRA DEL
694                                           ; DIVISORE DI 1 BIT
695  12D0  C203           LD    $D1+2(2)      ; DA EFFETTUARSI 4 VOLTE
696  12D2  F203           ADD   $D1+2(2)
697  12D4  CA03           ST    $D1+2(2)
698  12D6  C202           LD    $D1+1(2)
699  12D8  F202           ADD   $D1+1(2)
700  12DA  CA02           ST    $D1+1(2)
701  12DC  C201           LD    $D1(2)
702  12DE  F203           ADD   $D1+2(2)
703  12E0  CA03           ST    $D1+2(2)
704  12E2  BAF8           DLD   $TEMP (2)
705  12E4  9CE9           JNZ   $SL1
706  12E6  AAF9           ILD   $CNT (2)      ; CONTATORE = CONTATORE + 1
707  12E8  90DB           JMP   $SET
708  12EA  C400   $SETUP: LDI   0             ; QUOZIENTE = 0
709  12EC  CAFD           ST    $Q(2)
710  12EE  CAFE           ST    $Q+1(2)
711  12F0  CAFF           ST    $Q+2(2)
712  12F2  CA00           ST    0(2)
713  12F4  CA04           ST    4(2)
714  12F6  9002           JMP   $SUB          ; INIZIO DELLA SOTTRA-
715                                           ; ZIONE
716                   ;
717                   ;       INCREMENTO DEL QUOZIENTE
718                   ;       ILD PUÒ ESSERE USATA FINCHÉ NON CI SARÀ
719                   ;       UN CARRY DA UN DIGIT AL SUCCESSIVO
720  12F8  AAFF   $INCQ:  ILD   $Q+2(2)       ; INCREMENTO DEL
721                                           ; QUOZIENTE BCD
722  12FA  C411   SUB:    LDI   H(BCDSUB)     ; CALL DELLA BCDSUB
723  12FC  37             XPAH  3
724  12FD  C4E0           LDI   L(BCDSUB)-1   ;LA SOTTRAZIONE DA
725  12FF  33             XPAL  3             ; EFFETTUARSI È:
726                                           ; DIVIDENDO - DIVISORE
727  1300  3F             XPPC  3
728  1301  9000           JMP   ONN
729  1303  C204   ONN:    LD    SD2-1(2)      ; TEST SE IL RISULTATO
730                                           ; È POSITIVO
731  1305  98F1           JZ    SINCQ         ; LO È, INCREMENTO
732                                           ; DEL QUOZIENTE
733  1307  C411           LDI   H(BCDADD)     ; NON LO È, UTILIZZO
734                                           ; DELLA BCDADD
735  1309  37             XPAH  3
736  130A  C4F3           LDI   L(BCDADD)-1
737  130C  33             XPAL  3
738  130D  3F             XPPC  3
739  130E  9000           JMP   ON
740  1310  BAF9   ON:     DLD   $CNT(2)       ; CONTATORE = CONTATORE - 1
741  1312  982D           JZ    $DONE         ; SE IL CONTATORE È = 0
742                                           ; ALLORA LA DIVISIONE
743                                           ; ATTUATA
744  1314  C404           LDI   4
745  1316  CAF8           ST    $TEMP(2)
746  1318  C201   $SL2:   LD    $D1(2)        ; DIVISIONE DEL DIVISORE
747                                           ; PER 10
748  131A  02             CCL
749  131B  1F             RRL                 ; SHIFT A DESTRA DI 4
750  131C  CA01           ST    $D1(2)        ; BIT
```

```
751  131E  C202              LD     $D1+1(2)
752  1320  1F                RRL
753  1321  CA02              ST     $D1+1(2)
754  1323  C203              LD     $D1+2(2)
755  1325  1F                RRL
756  1326  CA03              ST     $D1+2(2)
757  1328  02                CCL
758  1329  C2FF              LD     $Q+2(2)
759  132B  F2FF              ADD    $Q+2(2)
760  132D  CAFF              ST     $Q+2(2)
761  132F  C2FE              LD     $Q+1(2)
762  1331  F2FE              ADD    $Q+1(2)
763  1333  CAFE              ST     $Q+1(2)
764  1335  C2FD              LD     $Q(2)
765  1337  F2FD              ADD    $Q(2)
766  1339  CAFD              ST     $Q(2)
767  133B  BAF8              DLD    $TEMP(2)
768  133D  9CD9              JNZ    $SL2
769  133F  90B9              JMP    $SUB
770                  ;
771  1341  C2FC    $DONE:    LD     $Q-1(2)     ; COPIA DEL QUOZIENTE
772  1343  CA04              ST     $T          SD2-1(2)
773  1345  C2FD              LD     $Q(2)
774  1347  CA05              ST     $D2(2)
775  1349  C2FE              LD     $Q+1(2)
776  134B  CA06              ST     $D2+1(2)
777  134D  C2FF              LD     $Q+2(2)
778  134F  CA07              ST     $D2+2(2)
779  1351  C2FA              LD     $P3(2)      ; RIPRISTINO DEL VALORE
780                                            ; ORIGINALE IN P3
781  1353  37                XPAH   3
782  1354  C2FB              LD     $P3+1(2)
783  1356  33                XPAL   3
784  1357  C702              LD     @2(3)       ; INCREMENTO DI P3
785                                            ; DI 2 PER IL RITORNO
786                                            ; NORMALE
787  1359  C604              LD     @4(2)       ; INCREMENTO DI P2 PER
788                                            ; PUNTARE AL QUOZIENTE
789  135B  3F      $OUT:     XPPC   3           ; RITORNO
790  135C  C412              LDI    H(BCDDIV)
791  135E  35                XPAH   1
792  135F  C4A5              LDI    L(BCDDIV)
793  1361  31                XPAL   1
794  1362  91FF              JMP    0(1)        ;JMP A BCDDIV
795                  ;
796                  ;
797  1364  C3FA    $ERR:     LD     $P3(2)      ; OVERFLOW IN BCDADD O
798                                            ; BCDSUB
799  1366  37                XPAH   3
800  1367  C2FB              LD     $P3+1(2)
801  1369  33                XPAL   3
802  136A  90EF              JMP    $OUT
803                  ;
804                  ;
805                  ;
806        0000              .END
```

6
I MICROPROCESSORI
E LE LORO APPLICAZIONI:
SC/MP
ALDO CAVALCOLI
VALERIO SCIBILIA
JACKSON ITALIANA EDITRICE